Translations between Modal Logics of Reactive Systems*

F. Laroussinie S. Pinchinat and Ph. Schnoebelen
Lifia-Imag, 46 Av. Félix Viallet, F-38000 Grenoble, France
Email : {fl,spinchi,phs}@lifia.imag.fr

September 20, 1994

Abstract

We propose meaning-preserving translations between L_B, L_U and L_{sb} (three modal logics in full agreement with branching bisimulation), thus proving that they all have the same expressivity. The translations can be implemented and have potential applications in the automated analysis of reactive systems.

In this work the main difficulty is that L_B uses both forward and backward modalities, whereas L_U and L_{sb} only have forward modalities. The technique we developed to cope with this, is an adaptation in a branching-time framework of the methods underlying Gabbay's Separation Theorem for PTL [8]. This technique is powerful and has been applied successfully to related problems.

1 Introduction

Modal logic is an important tool in the analysis, specification and verification of reactive systems [22]. Among many other applications, logics like the Hennessy-Milner Logic (shortly, HML) have been used as a benchmark for semantic equivalences [12], as the specification language used in model checking tools [2], and as a language in which to explain why two systems are not semantically equivalent [14]. A classical result of modal characterization of semantic equivalences, is the adequacy theorem of Hennessy and Milner stating that in a (finitely branching) transition system, two states p and q are bisimilar, written $p \underline{\leftrightarrow} q$, iff they satisfy the same HML formulae, written $p \equiv_{HML} q$, where

$$p \equiv_L q \overset{\text{def}}{\Leftrightarrow} \forall f \in L \, (p \models f \;\Leftrightarrow\; q \models f).$$

This fundamental result is a strong point in favor of bisimulation equivalence as the key semantic equivalence for CCS [17, 19]. It also helps to explain the concepts underlying bisimulation equivalence. Following the direction exemplified in [12], many other behavioral equivalences have been

*This article extends our earlier work presented in *Proc. AMAST'93, Enschede, NL, June 1993.*

characterized through modal logics: see [18, 1, 13, 21, 3, 5, 10] among many others.

Here, we are mostly interested in modal logics with past-time (backward) modalities. A few exist. They have been used (among other applications) to capture non-continuous properties of *generalized* transition systems (J_T in [13]), to characterize history-preserving bisimulation in causality-based models (L_P in [3]) and to characterize branching bisimulation by mimicking back-and-forth τ-bisimulation (L_B in [5]).
In particular, regarding L_B, we want to compare it (in terms of expressivity) with L_U and L_{sb}, two modal logics with only forward modalities, that also characterize branching bisimulation. The existing literature [5, 10] establishes that they have the same distinguishing power:

$$p \equiv_{L_B} q \text{ iff } p \equiv_{L_U} q \text{ iff } p \equiv_{L_{sb}} q$$

because, writing $\underline{\leftrightarrow}_b$ for branching bisimulation, $p \equiv_L q$ iff $p \underline{\leftrightarrow}_b q$ for any $L \in \{L_B, L_U, L_{sb}\}$.
Formally speaking, these results do not compare the *expressivity* of the L_B, L_U and L_{sb} logics. One usually says that two logics L and L' have the same expressivity when any formula of one logic has an equivalent (in some formal sense) in the other logic. (When the equivalent formula can be effectively computed, we say that there exists a *translation* algorithm.) While it is very common in other fields, this particular question has not received much attention in the field of modal logics for reactive systems. Regarding L_B, L_U and L_{sb}, this article shows, through *three translation theorems* of the general form $L \preceq L'$, that they can all be translated into any other. Our translation theorems use specific techniques we developed for branching-time temporal logics with past [16]. Usually, the main technical difficulty is to establish a so-called *separation theorem.*
Our motivations are not only theoretical. The translations we describe are constructive, easy to implement, and potentially useful in the automated analysis of reactive systems. For example, by showing how to translate HML_{bf} (HML with past-time connectives) into its future-time fragment HML, we show how to easily expand the input language of any software tool (e.g. a verifier) handling HML properties. Similarly, the translations between L_B, L_U and L_{sb} can be combined with the diagnostic mechanism of [15] (which uses L_U to explain why two systems are not branching-bisimilar) to offer explanations in different modal languages.

All the logics we consider in this article are variants of HML:

HML_{bf}: is a back-and-forth version of HML, in a framework with only visible labels,

L_U: is a version of HML with an "until" modality, in a framework with invisible labels (τ's),

L_{sb}: is a weaker L_U, inspired from the definition of semi-branching bisimulation,

L_B: is a version of HML_{bf} incorporating τ's.

In Section 2 we recall the technical framework (transition systems and modal logics with backward modalities) in a setting with no invisible (a.k.a. τ) labels. We discuss the expressivity and translation issues in this basic setting (Section 3) where it is already possible to give a first translation theorem (Section 4). Interesting in its own right, this theorem also has pedagogical virtues, as it exemplifies the approach we use in the remainder of the article. Then we move (Section 5) to systems with τ-steps and logics for branching bisimulation. We present a few preliminary results in Section 6 and establishes the three main translation theorems in Sections 7 and 8.

2 Logics with backward modalities

We consider a fixed set $A = \{a, b, \ldots\}$ of *labels*. A *labeled transition system* (LTS) is an edge-labeled graph $\langle Q, \rightarrow\rangle$ where $Q = \{p, q, \ldots\}$ is a set of *states* and $\rightarrow \subseteq Q \times A \times Q$ is the *transition relation*. We assume a fixed LTS S.

2.1 Syntax

HML_{bf} (read "HML back-and-forth") is HML augmented with past-tense (*backward*) modalities. It was introduced in [5] for systems with τ's (but observe that HML_{bf} is a subset of J_T defined in [13]).

Definition 2.1 ***(Syntax of*** HML_{bf}***)*** *HML_{bf} formulae are built according to the following grammar:*

$$HML_{bf} \ni f, g \ ::= \ \top \ | \ \neg f \ | \ f \wedge g \ | \ \langle a\rangle f \ | \ \overline{\langle a\rangle} f$$

where a is any action from A.

HML is the fragment of HML_{bf} where the $\overline{\langle a\rangle}$ modalities are not allowed.
We use $f, g, \alpha, \beta, \varphi, \psi, \ldots$ to denote HML_{bf} formulae and we use the standard abbreviations: $f \vee g$, $\perp$, $[a]f$ (for $\neg\langle a\rangle\neg f$) and $\overline{[a]}f$ (for $\neg\overline{\langle a\rangle}\neg f$).

2.2 Semantics

A modal logic with backward modalities states properties of a *run* $\pi = [q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n]$ of S. A run like π is a partial computation of S starting from a state q_0 and currently in state q_n. This partial computation can be expanded (if q_n is not a final state) and we write $\pi \xrightarrow{a} \pi'$ when run π' is π with a transition $q_n \xrightarrow{a} q_{n+1}$ added. If $n > 0$ the run has a past (a history) and the backward modalities in HML_{bf} can be used to state properties of this past.

Definition 2.2 ***(Semantics of*** HML_{bf}***)*** *For a run* π *of some LTS* S *and an* HML_{bf} *formula* f*, we define when* $\pi \models_S f$ *(reads "*π *satisfies* f*") by induction on the structure of* f*:*

$$\begin{array}{ll} \pi \models \top & \textit{always,} \\ \pi \models \neg f & \textit{iff } \pi \not\models f, \\ \pi \models f \wedge g & \textit{iff } \pi \models f \textit{ and } \pi \models g, \\ \pi \models \langle a \rangle f & \textit{iff there is a } \pi \xrightarrow{a} \pi' \textit{ s.t. } \pi' \models f, \\ \pi \models \overline{\langle a \rangle} f & \textit{iff there is a } \pi' \xrightarrow{a} \pi \textit{ s.t. } \pi' \models f. \end{array}$$

(The "S"subscript is omitted whenever no confusion can arise.) In this framework, there is some asymmetry between past and future because (1) past is finite, while future needs not be, and (2) past is "deterministic", or fixed by the history, while future is branching.

3 Equivalent formulae and translations between logics

In practice, we use HML_{bf} to express properties of states (mainly the initial state of the system) rather than runs. For a state $q \in Q$, the derived notion $q \models f$ is given by

$$q \models f \stackrel{\text{def}}{\Leftrightarrow} [q] \models f$$

$[q]$ is just state q seen as a run, with no past. We say that states p and q satisfy the same HML_{bf} formulae, written $p \equiv_{HML_{bf}} q$, when $p \models f \Leftrightarrow q \models f$ for all $f \in HML_{bf}$.
[5] mention that $p \equiv_{HML_{bf}} q$ iff $p \underline{\leftrightarrow} q$ because (strong) bisimulation coincides with (strong) back-and-forth bisimulation [4]. This entails

$$p \equiv_{HML} q \text{ iff } p \equiv_{HML_{bf}} q \tag{1}$$

In the following, we are looking for a finer comparison between the expressive powers of HML and HML_{bf}. We consider whether formulae of HML_{bf} can be translated into HML. Of course, a formula like $\overline{\langle a \rangle}\top$, which says that the last step was an a-step cannot be written in HML where only properties about the possible futures can be expressed. But when we express properties of states (without a past), we know that we never have $q \models \overline{\langle a \rangle}\top$. Thus, in a certain sense, $\overline{\langle a \rangle}\top$ (an HML_{bf} formula) can be correctly translated into $\perp$ (an HML formula).
This requires some definitions:

Definition 3.1 *Two formulae are* globally equivalent*, written* $f \equiv f'$*, iff* $\pi \models f \Leftrightarrow \pi \models f'$ *for all runs* π *in all LTS's.*
They are initially equivalent*, written* $f \equiv_i f'$*, iff* $q \models f \Leftrightarrow q \models f'$ *for all states* q *in all LTS's.*

For example, we have $\overline{\langle a \rangle}\top \equiv_i \perp$ but $\overline{\langle a \rangle}\top \not\equiv \perp$. Clearly, $f \equiv f'$ implies $f \equiv_i f'$ but the converse is not true as seen above.

When we just say "equivalent", we mean "globally equivalent". Global equivalence is the natural notion of equivalence on formulae [7]. It is a congruence: if $f \equiv f'$ with f a subformula of h (that is, h is some $h[f]$) then $h \equiv h[f']$. This does not hold for $\equiv_i$ which is only a congruence w.r.t. boolean combinators and backward modalities.

Now we can define what is a translation between two logics.

Definition 3.2 *A logic L can be translated (resp. initially translated) into L', written $L \preceq_g L'$ (resp. $L \preceq_i L'$), iff for any $f \in L$ there is a $f' \in L'$ with $f \equiv f'$ (resp. $f \equiv_i f'$).*

Clearly, $L \preceq_g L'$ implies $L \preceq_i L'$. Also $L \preceq_i L'$ implies $\equiv_{L'} \subseteq \equiv_L$. In both cases, the reverse implication is not true in general.
One trivial example is $HML \preceq_g HML_{bf}$, which holds because $HML \subseteq HML_{bf}$. We now investigate the reverse direction.

4 From HML_{bf} to HML

Theorem 4.1 *$HML_{bf} \preceq_i HML$.*

The proof is in two steps: we first "separate" HML_{bf} formulae modulo $\equiv$, and then translate separated formulae into initially equivalent HML formulae. This requires some preparation.
Say a formula is *pure-past* (resp. *pure-future*) if it does not contain forward (resp. backward) modalities. Say it is *separated* if no backward modality occurs in the scope of a forward modality (and write HML_{bf}^{sep} for the fragment of HML_{bf} that contains only separated formulae).

Proposition 4.2 ***(Separation Lemma for** HML_{bf}**)***

$$HML_{bf} \preceq_g HML_{bf}^{sep}. \tag{2}$$

Proof We show that any f in HML_{bf} is equivalent to a separated f'. The proof is by structural induction on f, . The cases when f has the form $\top$, $g_1 \wedge g_2$, or $\neg g$ are obvious.

$f = \overline{\langle a \rangle} g$: g can be separated (by ind. hyp.) into some g'. Then $f \equiv f' \stackrel{\text{def}}{=} \overline{\langle a \rangle} g'$ is separated.

$f = \langle a \rangle g$: g can be separated (by ind. hyp.) into some g'. There are two subcases:

– Assume g' has the form $\overline{\langle b_1 \rangle}\varphi_1 \wedge \cdots \wedge \overline{\langle b_n \rangle}\varphi_n \wedge \neg\overline{\langle c_1 \rangle}\varphi'_1 \wedge \cdots \wedge \neg\overline{\langle c_m \rangle}\varphi'_m \wedge \psi^+$ where ψ^+ is pure future. Write $c_{i_1}, \ldots, c_{i_k}$ for the c_i's that are equal to a. Then

$$\langle a \rangle g' \quad \equiv \quad g'' \quad \stackrel{\text{def}}{=} \begin{cases} \varphi_1 \wedge \cdots \wedge \varphi_n \wedge \neg\varphi_{i_1} \wedge \cdots \wedge \neg\varphi_{i_k} \wedge \langle a \rangle \psi^+ & \text{if } b_i = a \text{ for } i = 1, \ldots, n, \\ \bot & \text{otherwise.} \end{cases}$$

$f \equiv g''$ and g'' is separated.

– In the general case, g' can be put in disjunctive normal form $\bigvee_i \bigwedge_j g_{i,j}$ where every $g_{i,j}$ has the form $\langle a\rangle\varphi$, $\neg\langle a\rangle\varphi$, $\overline{\langle a\rangle}\varphi$ or $\neg\overline{\langle a\rangle}\varphi$. The $g_{i,j}$'s are separated. $f \equiv \langle a\rangle g' \equiv \bigvee_i \langle a\rangle\Big(\bigwedge_j g_{i,j}\Big)$ and each $\langle a\rangle\Big(\bigwedge_j g_{i,j}\Big)$ falls in the previous subcase and can be separated.

$\square$

Remark 4.3 In a linear-time framework, [8, 9] use a different, less general, definition of separated formulae: a formula is *separated (in Gabbay's sense)* if it is a boolean combination of pure-past and pure-future formulae. Our definition is required in branching-time frameworks (see [16]). For example (2) does not hold for Gabbay's definition of separated formulae: $\overline{\langle a\rangle}\langle b\rangle\top$ has no equivalent as a boolean combination of pure-past and pure-future HML_{bf} formulae.

Now we conclude the proof of Theorem 4.1 with

Proposition 4.4 $HML_{bf}^{sep} \preceq_i HML$.

Proof Use $\overline{\langle a\rangle}f \equiv_i \bot$ to eliminate (modulo $\equiv_i$) all backward modalities since they are not in the scope of a forward modality. $\square$

5 Modal logics for branching-bisimulation

We now move to a setting where invisible steps are allowed. Such steps are a fundamental way of modeling the abstraction operation required for the hierarchical description of systems [17, 19]. We write τ for this invisible label and consider transition systems labeled over $A_\tau \stackrel{\text{def}}{=} A \cup \{\tau\}$. We write $q \Rightarrow q'$ when there is a sequence $q \xrightarrow{\tau} \ldots \xrightarrow{\tau} q'$. That is, $\Rightarrow$ is the transitive and reflexive closure of $\xrightarrow{\tau}$. In this setting, a very natural equivalence is *branching bisimulation* [11, 10]. [5] introduces L_U and L_B, two modal logics characterizing branching bisimulation.

5.1 L_B

L_B is a version of HML_{bf} adapted to systems with invisible moves.

Definition 5.1 ***(Syntax of*** L_B***)*** *The formulae of* L_B *are built according to the following grammar*

$$L_B \ni f, g \ ::= \ \top \ \mid \ \neg f \ \mid \ f \wedge g \ \mid \ \langle\langle k\rangle\rangle f \ \mid \ \overline{\langle\langle k\rangle\rangle} f$$

where k *is any label from* $A_\epsilon \stackrel{def}{=} A \cup \{\epsilon\}$.

We use $[[k]]f$ and $\overline{[[k]]}f$ as standard abbreviations.

The semantics of the new modalities is given by:

Definition 5.2 ***(Semantics of L_B)***

$$
\begin{array}{ll}
\pi \models \langle\langle a\rangle\rangle f & \textit{iff there is a } \pi \Rightarrow\xrightarrow{a}\Rightarrow \pi' \textit{ s.t. } \pi' \models f,\\
\pi \models \langle\langle \epsilon\rangle\rangle f & \textit{iff there is a } \pi \Rightarrow \pi' \textit{ s.t. } \pi' \models f,\\
\pi \models \overline{\langle\langle a\rangle\rangle} f & \textit{iff there is a } \pi' \Rightarrow\xrightarrow{a}\Rightarrow \pi \textit{ s.t. } \pi' \models f,\\
\pi \models \overline{\langle\langle \epsilon\rangle\rangle} f & \textit{iff there is a } \pi' \Rightarrow \pi \textit{ s.t. } \pi' \models f.
\end{array}
$$

Clearly, the inspiration behind L_B is the definition of *back-and-forth weak bisimulation* [4], which coincides with branching bisimulation.
Beside boolean manipulations, we often use the following basic equivalences between L_B formulae:

Lemma 5.3 *For all $f, \ldots$ in L_B and $k \in A_\epsilon$,*

1. $\langle\langle k\rangle\rangle(\bigvee_i f_i) \equiv \bigvee_i(\langle\langle k\rangle\rangle f_i)$,
2. $\overline{\langle\langle k\rangle\rangle}(\bigvee_i f_i) \equiv \bigvee_i(\overline{\langle\langle k\rangle\rangle} f_i)$,
3. $\langle\langle k\rangle\rangle\langle\langle \epsilon\rangle\rangle f \equiv \langle\langle \epsilon\rangle\rangle\langle\langle k\rangle\rangle f \equiv \langle\langle k\rangle\rangle f$,
4. $\overline{\langle\langle k\rangle\rangle\langle\langle \epsilon\rangle\rangle} f \equiv \overline{\langle\langle \epsilon\rangle\rangle\langle\langle k\rangle\rangle} f \equiv \overline{\langle\langle k\rangle\rangle} f$,
5. $\langle\langle a\rangle\rangle\overline{\langle\langle \epsilon\rangle\rangle} f \equiv \langle\langle a\rangle\rangle f$,
6. $\overline{\langle\langle a\rangle\rangle}\langle\langle \epsilon\rangle\rangle f \equiv \overline{\langle\langle a\rangle\rangle} f$,
7. $\langle\langle \epsilon\rangle\rangle\overline{[[k]]} f \equiv \overline{[[k]]} f$.

5.2 L_U

L_U has no backward modalities but it has a so-called "until" modality which extends the simple forward modality of L_B.

Definition 5.4 ***(Syntax of L_U)*** *The formulae of L_U are built according to the following grammar*

$$L_U \ni f, g \ ::= \ \top \ | \ \neg f \ | \ f \wedge g \ | \ f\langle k\rangle g$$

with $k \in A_\epsilon$.

The semantics is given by

Definition 5.5 ***(Semantics of L_U)***

$$
\begin{array}{ll}
\pi \models f\langle a\rangle g & \textit{iff } \exists n > 0,\ \pi = \pi_0 \xrightarrow{\tau} \pi_1 \xrightarrow{\tau} \cdots \pi_{n-1} \xrightarrow{a} \pi_n \textit{ s.t. } \pi_n \models g \textit{ and } \pi_i \models f \textit{ for } i < n,\\
\pi \models f\langle \epsilon\rangle g & \textit{iff } \exists n \geq 0,\ \pi = \pi_0 \xrightarrow{\tau} \pi_1 \xrightarrow{\tau} \cdots \pi_{n-1} \xrightarrow{\tau} \pi_n \textit{ s.t. } \pi_n \models g \textit{ and } \pi_i \models f \textit{ for } i < n.
\end{array}
$$

Then, the L_U formula $f\langle a\rangle g$ requires that f hold continuously until some moment when g will be true immediately after an a step. The inspiration behind L_U is the definition of branching bisimulation [11]. L_U's "until" modality is stronger than L_B's forward modalities. Indeed, we have:

$$\langle\langle k\rangle\rangle f \equiv \top\langle k\rangle(\top\langle\epsilon\rangle f). \tag{3}$$

while we don't see any way of expressing "until" as a combination of $\langle\langle.\rangle\rangle$ and $\overline{\langle\langle.\rangle\rangle}$ (and believe that no solution exists).
The only distributive property of "until" is

$$f\langle k\rangle(g_1 \vee g_2) \equiv (f\langle k\rangle g_1) \vee (f\langle k\rangle g_2) \tag{4}$$

5.3 L_{sb}

Van Glabbeek proposed a weaker version of an "until" modality that does not express *continuous* copying [10].

Definition 5.6 ***(Syntax of*** L_{sb}***)*** *The formulae of* L_{sb} *are built according to the following grammar*

$$L_{sb} \ni f, g \ ::= \ \top \ \mid \ \neg f \ \mid \ f \wedge g \ \mid \ f\{k\}g$$

with $k \in A_\epsilon$.

Definition 5.7 ***(Semantics of*** L_{sb}***)***

$$\begin{array}{ll} \pi \models f\{a\}g & \textit{iff there is a } \pi \Rightarrow \pi' \xrightarrow{a} \pi'' \textit{ s.t. } \pi'' \models g \textit{ and } \pi' \models f, \\ \pi \models f\{\epsilon\}g & \textit{iff there is a } \pi \Rightarrow \pi' \textit{ s.t. } \pi' \models f \textit{ and } (\pi' \models g \textit{ or there is a } \pi' \xrightarrow{\tau} \pi'' \textit{ with } \pi'' \models g). \end{array}$$

Clearly, the inspiration behind L_{sb} is the definition of *semi branching bisimulation* [11], which coincides with branching bisimulation. When $\pi \models f\{a\}g$, we do not state any property of the intermediary states between π and π'. This gives technical simplicity: in order to satisfy $f\{k\}g$, it is only necessary to satisfy f in one future place. This explains why

$$(f \vee f')\{k\}g \equiv f\{k\}g \ \vee \ f'\{k\}g \tag{5}$$

is valid. L_U offers no such property. Clearly, L_{sb} is weaker than L_U and indeed L_{sb} is readily translated into L_U through

$$f\{k\}g \equiv \langle\langle\epsilon\rangle\rangle(f \ \wedge \ f\langle k\rangle g) \tag{6}$$

entailing $L_{sb} \subseteq_g L_U$.

5.4 L_{BU}

For technical reasons, we introduce L_{BU} [23], a logic built by combining all modalities of L_U and L_B (and L_{sb}), so that all three logics are fragments of a common superset:

$$L_{BU} \ni f, g \ ::= \ \top \ | \ \neg f \ | \ f \wedge g \ | \ \langle\langle k \rangle\rangle f \ | \ \overline{\langle\langle k \rangle\rangle} f \ | \ f \langle k \rangle g \ | \ f \{k\} g$$

with $k \in A_\epsilon$. (Clearly, some modalities are redundant in L_{BU} because of (3) and (6).)
We can then use generic concepts for our three modal logics by just referring to L_{BU}. For example the modal height of a formula is defined (as the maximal number of nested modalities) for all L_{BU} formulae.

Considering that $\equiv_{L_U}$, $\equiv_{L_B}$ and $\equiv_{L_{sb}}$ (and $\equiv_{L_{BU}}$) coincide, a natural question is whether any of the three logics can be translated into each other. This question has already been addressed for L_U and L_B [6, 23] but complete answers have not yet been offered.
The rest of the article is devoted to a proof that $L_U \preceq_g L_{sb} \preceq_g L_B$ and $L_B \preceq_i L_U$. Using L_{sb} as an intermediary logic between L_U and L_B greatly simplified our earlier proof.

6 $\Diamond$- and $\Box$-formulae

This section develops some useful concepts for the following sections. The aim is to study a specific class of formulae which behave well in the left-hand sides of L_U's "until" modalities in the sense that they enjoy distributivity properties not satisfied by arbitrary formulae.

Definition 6.1 *An L_{BU} formula f is a* $\Diamond$-formula *iff for all π, π' in all LTS's, $\pi \models f$ and $\pi' \Rightarrow \pi$ imply $\pi' \models f$. It is a* $\Box$-formula *iff for all π, π' in all LTS's, $\pi \models f$ and $\pi \Rightarrow \pi'$ imply $\pi' \models f$.*

Thus when a $\Box$-formula (resp. $\Diamond$-formula) holds of some π, it holds in all τ-successors (resp. τ-predecessors) of π. This is why for any $\Box$-formulae $f^\Box$ and $g^\Box$ and any $\Diamond$-formulae $f^\Diamond$ and $g^\Diamond$

$$\begin{aligned}(f^\Box \vee g^\Box)\langle k \rangle h \ &\equiv \ (f^\Box \langle k \rangle h) \vee (g^\Box \langle k \rangle h)\\ (f^\Diamond \vee g^\Diamond)\langle k \rangle h \ &\equiv \ (f^\Diamond \langle k \rangle h) \vee (g^\Diamond \langle k \rangle h)\end{aligned}$$

We write informally $f \in \Diamond$ (resp. $f \in \Box$) when f is a $\Diamond$-formula (resp. a $\Box$-formula). A given formula may well be both a $\Box$- and a $\Diamond$-formula (witness $\top$ and $\bot$) or none.
The following properties are useful:

Lemma 6.2 *For all $f, g \in L_{BU}$ and all $k \in A_\epsilon$,*

1. *$f \in \Diamond$ iff $\neg f \in \Box$,*
2. *$f \in \Box$ iff $\neg f \in \Diamond$,*
3. *$f, g \in \Diamond$ implies $f \wedge g, f \vee g \in \Diamond$,*

4. $f, g \in \Box$ *implies* $f \wedge g, f \vee g \in \Box$,

5. $f \in \Diamond$ *iff* $f \equiv \langle\langle\epsilon\rangle\rangle f$,

6. $f \in \Box$ *iff* $f \equiv [[\epsilon]] f$,

7. $\langle\langle k\rangle\rangle f, \overline{[[k]]} f \in \Diamond$,

8. $[[k]] f, \overline{\langle\langle k\rangle\rangle} f \in \Box$,

9. $f\{k\}g \in \Diamond$.

Proof *1.-4.* clear from the definition.
5. A simple exercise left to the reader.
6. Combine *2.* and *5.*
7. Combine *5.* and Lemma 5.3, points *3.* and *7.* Use duality to prove *8.*
9. Combine (6) and *7.* $\Box$

Points *5.* and *6.* above may help understand our choice of terminology. With Lemma 6.2.*9* above, we have the important

Corollary 6.3 *Any* $f \in L_{sb}$ *is a boolean combination of* $\Diamond$*-formulae in* L_{sb}.

A similar result is true for L_B also (witness Lemma 6.2.*7*) but not for L_U (witness $(\neg \top\langle a\rangle\top)\langle b\rangle\top$).

7 From L_U to L_B

All L_U formulae (in fact, all L_{BU} formulae, see Theorem 8.11) can be translated into L_B. In this section, we show how to go from L_U to L_{sb} and then from L_{sb} to L_B.

Theorem 7.1 $L_U \preceq_g L_{sb}$.

Proof We show, by structural induction on f, that any $f \in L_U$ can be translated into an equivalent formula in L_{sb}. The interesting case is when f is some $g\langle k\rangle h$. Then, by ind. hyp., g and h can be translated into g' and h' in L_{sb}. Using Corollary 6.3, we can write g' in disjunctive normal form and assume

$$f \equiv \Big(\bigvee_{i=1}^{n} (f_i^\Diamond \wedge g_i^\Box)\Big)\,\langle k\rangle\, h'$$

where, for $i = 1, \ldots, n$, $f_i^\Diamond \in \Diamond$ and $g_i^\Box \in \Box$. We now reason by induction on n.

- First consider the simpler case where $n = 1$. We use

$$(f^\Diamond \wedge g^\Box)\langle a\rangle h' \equiv g^\Box \wedge (f^\Diamond\{a\}h') \tag{7}$$

$$(f^\Diamond \wedge g^\Box)\langle \epsilon\rangle h' \equiv \Big(g^\Box \wedge (f^\Diamond\{\epsilon\}h')\Big) \vee h' \tag{8}$$

  and immediately obtain L_{sb} formulae.

- Now in the general case where $n > 1$, we use

$$\Big(\bigvee_{i=1}^{n}(f_i^\diamond \wedge g_i^\Box)\Big)\,\langle a\rangle\, h' \;\equiv\; \bigvee_{j=1}^{n}\Bigg[g_j^\Box \wedge \Big(f_j^\diamond\{a\}h' \vee f_j^\diamond\{\epsilon\}\Big(\Big(\bigvee_{\substack{i=1\\i\neq j}}^{n}(f_i^\diamond \wedge g_i^\Box)\Big)\langle a\rangle h'\Big)\Big)\Bigg] \tag{9}$$

$$\Big(\bigvee_{i=1}^{n}(f_i^\diamond \wedge g_i^\Box)\Big)\,\langle \epsilon\rangle\, h' \;\equiv\; h' \vee \bigvee_{j=1}^{n}\Bigg[g_j^\Box \wedge f_j^\diamond\{\epsilon\}\Big(\Big(\bigvee_{\substack{i=1\\i\neq j}}^{n}(f_i^\diamond \wedge g_i^\Box)\Big)\langle \epsilon\rangle h'\Big)\Bigg] \tag{10}$$

  which can be translated by ind. hyp.

We let the reader check that (7-10) hold when $f_i^\diamond \in \diamond$ and $g_i^\Box \in \Box$ for all i. As an indication, we can give the intuition behind (9): assume $\pi \models \Big(\bigvee_{i=1}^{n}(f_i^\diamond \wedge g_i^\Box)\Big)\langle a\rangle\, h$. Then there is a path $\pi = \pi_0 \cdots \xrightarrow{\tau} \pi_r \xrightarrow{a} \pi'$ with $\pi' \models h$ and s.t. any π_s $(0 \leq s \leq r)$ satisfies one of the $f_i^\diamond \wedge g_i^\Box$'s. In particular, $\pi \models f_j^\diamond \wedge g_j^\Box$ for some j (and then $\pi_s \models g_j^\Box$ for $s = 0, \ldots, r$). Now there are two cases:

- either $\pi_0, \ldots, \pi_r$ all satisfy $f_j^\diamond \wedge g_j^\Box$, and then $\pi \models g_j^\Box \wedge (f_j^\diamond\{a\}h)$, as for (7),
- or there is a $0 < s \leq r$ s.t. $\pi_s \not\models f_j^\diamond$. We pick the smallest such s. Then, because $f_j^\diamond \in \diamond$, none of $\pi_s, \pi_{s+1}, \ldots, \pi_r$ satisfies $f_j^\diamond$. Therefore they all satisfy $\Big(\bigvee_{i\neq j}(f_i^\diamond \wedge g_i^\Box)\Big)\langle a\rangle h$.

$\Box$

Theorem 7.2 $L_{sb} \preceq_g L_B$.

Proof We show that any $f \in L_{sb}$ can be translated into an equivalent formula in L_B. This is done by induction on the modal height of f, and then by structural induction on f.
The interesting case is when f is some $g\{k\}h$. We know (Corollary 6.3) that g is a boolean combination of $\diamond$-formulae. Then, thanks to (5) and Lemma 6.2, it is enough to only consider formulae of the general form $(f^\diamond \wedge g^\Box)\{k\}h$, with $f^\diamond \in \diamond$ and $g^\Box \in \Box$. We use

$$(f^\diamond \wedge g^\Box)\{a\}h \;\equiv\; \langle\langle a\rangle\rangle\Big(\overline{[[\epsilon]]}h \wedge \overline{[[a]]}f^\diamond \wedge \overline{\langle\langle a\rangle\rangle}g^\Box\Big) \tag{11}$$

$$(f^\diamond \wedge g^\Box)\{\epsilon\}h \;\equiv\; \langle\langle \epsilon\rangle\rangle\Big[f^\diamond \wedge g^\Box \wedge \langle\langle \epsilon\rangle\rangle\Big(h \wedge \overline{[[\epsilon]]}(h \vee f^\diamond)\Big)\Big] \tag{12}$$

and there only remains to replace $f^\diamond$, $g^\Box$ and h by their L_B equivalent. (Again, we let the reader check that (11-12) are valid whenever $f \in \diamond$, $g \in \Box$.) $\Box$

Corollary 7.3 $L_U \preceq_g L_B$.

8 From L_B to L_U

The problem of translating L_B into L_U was considered in [23] where a partial solution is proposed. Our approach was developed independently and uses the separation techniques we exemplified in Section 4. This section establishes the following

Theorem 8.1 $L_B \preceq_i L_U$.

as a corollary of

Proposition 8.2 ***(Separation Lemma for*** L_{BU}***)***

$$L_{BU} \preceq_g L_{BU}^{sep}. \tag{13}$$

where L_{BU}^{sep} denotes the set of separated L_{BU} formulae, i.e. of formulae with no backward modality under the scope of a forward modality. The proof of Proposition 8.2 uses a set of valid equalities over L_{BU} formulae that are gathered in the following lemma. These equalities are sufficient to rewrite any L_{BU} formula into an equivalent separated formula.

Lemma 8.3 *For all* L_{BU} *formulae* $\alpha, \beta, \varphi, \varphi', \psi, \ldots$ *and labels* $a, b \in A$*,* $k \in A_\epsilon$*, we have*

$$\alpha\langle a\rangle(\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \beta) \equiv \alpha\langle a\rangle(\psi \wedge \beta) \tag{14}$$

$$\alpha\langle a\rangle(\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \beta) \equiv \alpha\langle a\rangle(\neg\psi \wedge \beta) \tag{15}$$

$$\alpha\langle\epsilon\rangle(\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \beta) \equiv (\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \alpha\langle\epsilon\rangle\beta) \ \vee \ \alpha\langle\epsilon\rangle(\psi \wedge \alpha\langle\epsilon\rangle\beta) \tag{16}$$

$$\alpha\langle\epsilon\rangle(\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \beta) \equiv \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \ \wedge \ (\alpha \wedge \neg\psi)\langle\epsilon\rangle(\beta \wedge \neg\psi) \tag{17}$$

$$\alpha\langle a\rangle(\overline{\langle\langle b\rangle\rangle}\psi \wedge \beta) \equiv \begin{cases} \bot & \text{if } a \neq b, \\ (\alpha\langle\epsilon\rangle(\psi \wedge \alpha\langle a\rangle\beta)) \ \vee \ (\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \alpha\langle a\rangle\beta) & \text{if } a = b. \end{cases} \tag{18}$$

$$\alpha\langle a\rangle(\neg\overline{\langle\langle b\rangle\rangle}\psi \wedge \beta) \equiv \begin{cases} \alpha\langle a\rangle\beta & \text{if } a \neq b, \\ \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \ \wedge \ (\alpha \wedge \neg\psi)\langle a\rangle\beta & \text{if } a = b. \end{cases} \tag{19}$$

$$\alpha\langle\epsilon\rangle(\overline{\langle\langle b\rangle\rangle}\psi \wedge \beta) \equiv \overline{\langle\langle b\rangle\rangle}\psi \wedge \alpha\langle\epsilon\rangle\beta \tag{20}$$

$$\alpha\langle\epsilon\rangle(\neg\overline{\langle\langle b\rangle\rangle}\psi \wedge \beta) \equiv \neg\overline{\langle\langle b\rangle\rangle}\psi \wedge \alpha\langle\epsilon\rangle\beta \tag{21}$$

$$\Big((\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle k\rangle\alpha \equiv \begin{array}{l} \overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha \\ \vee \ \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\neg\psi \wedge (\varphi' \vee \beta))\langle k\rangle\alpha \\ \vee \ \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\neg\psi \wedge (\varphi' \vee \beta))\langle\epsilon\rangle(\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha) \end{array} \tag{22}$$

$$\Big((\overline{\langle\langle b\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle b\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle k\rangle\alpha \equiv \Big(\overline{\langle\langle b\rangle\rangle}\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha\Big) \vee \Big(\neg\overline{\langle\langle b\rangle\rangle}\psi \wedge (\varphi' \vee \beta)\langle k\rangle\alpha\Big) \tag{23}$$

$$\begin{array}{l} \Big((\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle\epsilon\rangle\ (\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \alpha) \\ \qquad \equiv \overline{\langle\langle\epsilon\rangle\rangle}\psi \ \wedge \ (\varphi \vee \beta)\langle\epsilon\rangle\alpha \ \ \vee \ \ \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \Big(\neg\psi \wedge (\varphi' \vee \beta)\Big)\langle\epsilon\rangle\Big(\psi \ \wedge \ (\varphi \vee \beta)\langle\epsilon\rangle\alpha\Big) \end{array} \tag{24}$$

$$\Big((\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle\epsilon\rangle\ (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \alpha) \equiv \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \ \wedge \ \Big(\neg\psi \wedge (\varphi' \vee \beta)\Big)\langle\epsilon\rangle(\neg\psi \wedge \alpha) \tag{25}$$

Proof All equalities are proved by case analysis, considering the different ways a given formula may be satisfied by a given run. We just give a detailed proof of (22), the most complex equality, and leave the other proofs to the reader.

We first prove the "$\Rightarrow$" direction. For this, assume that $\pi \models \Big((\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle k\rangle\alpha$. For simplicity assume k is some visible a. Then there is some $\pi = \pi_o \xrightarrow{\tau} \pi_1 \xrightarrow{\tau} \cdots \pi_{n-1} \xrightarrow{a} \pi_n$ s.t.

$\pi_n \models \alpha$ and $\pi_i \models (\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta$ for all $i < n$. We distinguish three cases (illustrated in Figure 1):

Case 1: If $\pi_0 \models \overline{\langle\langle\epsilon\rangle\rangle}\psi$, then all π_i's ($0 \leq i < n$) satisfy $\overline{\langle\langle\epsilon\rangle\rangle}\psi$ and then must satisfy $\varphi \vee \beta$, so that $\pi \models \overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha$.

Case 2: Otherwise $\pi_0 \models \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi$.

Case 2.1: Assume all π_i's ($0 \leq i < n$) satisfy $\neg\psi$. Then they all satisfy $\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi$ and then must all satisfy $\varphi' \vee \beta$. So that $\pi \models \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\neg\psi \wedge (\varphi' \vee \beta))\langle k\rangle\alpha$.

Case 2.2: Otherwise the π_i's satisfy $\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi$ for all $i = 0, \ldots, m-1$ where π_m (with $0 < m < n$) is the first run in the sequence to satisfy ψ. Then the remaining π_i's ($m \leq i < n$) satisfy $\overline{\langle\langle\epsilon\rangle\rangle}\psi$. In this case, π_i must satisfy $\varphi' \vee \beta$ if $0 \leq i < m$, or $\varphi \vee \beta$ if $m \leq i < n$. Because $\pi_m \models \psi$, we have $\pi \models \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\neg\psi \wedge (\varphi' \vee \beta))\langle\epsilon\rangle(\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha)$.

Clearly, these three cases cover all possibilities. If now we assume $k = \epsilon$, the same reasoning applies except that there is one more possibility: π may satisfy the left-hand side of (22) by satisfying α. In this case $\pi \models \zeta\langle k\rangle\alpha$ for any ζ. As it also satisfies $\overline{\langle\langle\epsilon\rangle\rangle}\psi \vee \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi$, it must satisfy the disjunction $\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\varphi \vee \beta)\langle k\rangle\alpha \;\bigvee\; \neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge (\neg\psi \wedge (\varphi' \vee \beta))\langle k\rangle\alpha$ and then the right hand-side of (22).
Now, it should be clear that if π satisfies the right-hand side of (22), then we are necessarily in one of these three (or four, if $k = \epsilon$) cases, so that $\pi \models \Big((\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}\psi \wedge \varphi') \vee \beta\Big)\langle k\rangle\alpha$. $\square$

We can now turn to the separation theorem for L_{BU}, that is we describe how equalities (14) to (25) allow to rewrite any L_{BU} formula into an equivalent separated formula. Basically, (14) to (25) are sufficient to pull out any occurrence of a backward modality from the (immediate) scope of a forward modality. But this may bury other subformulae under several layers of forward modalities. Therefore the main difficulty is to find a strategy ensuring termination. For this we use an approach inspired from [8, 16]. The rewriting strategy is decomposed into a succession of lemmas dealing with more and more general cases.
As a technical simplification, we consider in this section that "until" is the only forward combinator in L_{BU}, thanks to (3) and (6). We also use L_{BU} *contexts*, that is L_{BU} formulae with variables serving as place-holders. Typically $f[x]$ denotes a context f where x may occur (possibly several times). Then $f[\varphi]$ is the L_{BU} formula obtained by replacing all occurrences of x with φ in $f[x]$. We write $f[x_1, \ldots, x_n] \equiv g[x_1, \ldots, x_n]$ when $f[\varphi_1, \ldots, \varphi_n] \equiv g[\varphi_1, \ldots, \varphi_n]$ for all $\varphi_1, \ldots \in L_{BU}$. The notions of "pure-future", "separated", ..., formula directly extend to contexts.

Lemma 8.4 *If $f[x]$ is a pure-future L_{BU} context, then $f[\overline{\langle\langle\epsilon\rangle\rangle}x]$ is equivalent to some separated $f'[x, \overline{\langle\langle\epsilon\rangle\rangle}x]$ with $f'[x, y]$ pure-future.*

Proof By structural induction on $f[x]$. The only interesting case is when $f[x]$ is an until-formula (that is, of the form $f_1[x]\langle k\rangle f_2[x]$). By ind. hyp., there are pure-future $f_1'[x, y]$ and $f_2'[x, y]$ s.t. $f[\overline{\langle\langle\epsilon\rangle\rangle}x]$

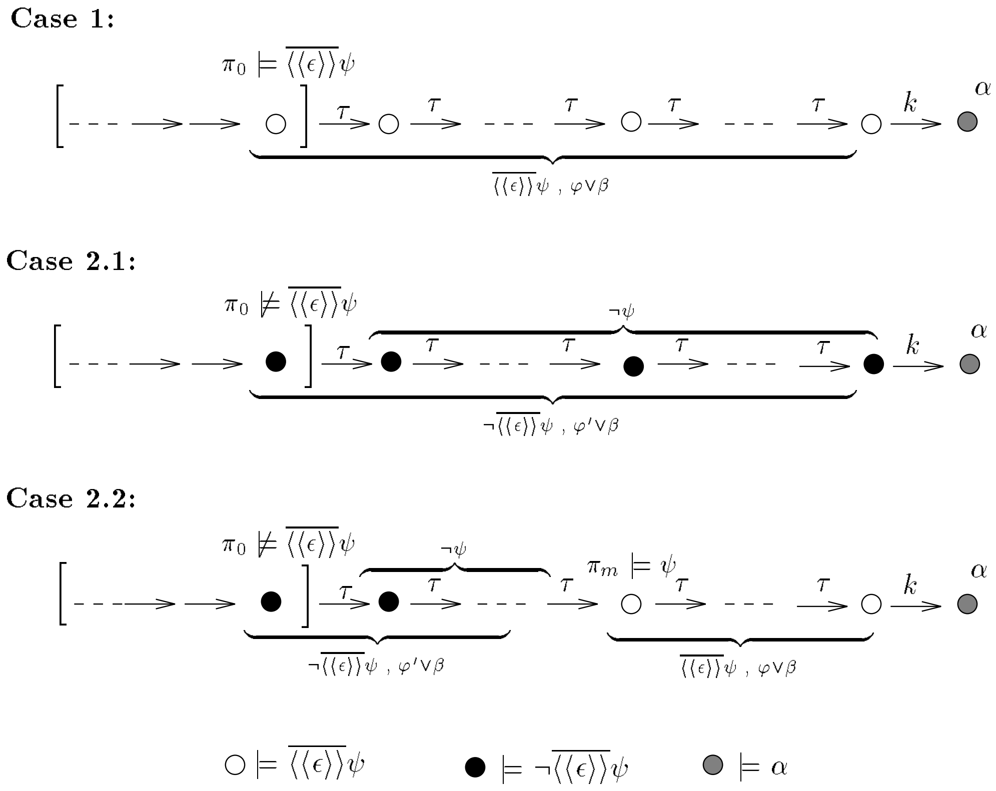

Figure 1: Three cases in the proof of (22)

is equivalent to $f'_1[x, \overline{\langle\langle\epsilon\rangle\rangle}x]\langle k\rangle f'_2[x, \overline{\langle\langle\epsilon\rangle\rangle}x]$, which we denote by $f''[x]$. Because the $f'_i[x, \overline{\langle\langle\epsilon\rangle\rangle}x]$'s are separated for $i = 1, 2$, all occurrences of $\overline{\langle\langle\epsilon\rangle\rangle}x$ in $f''[x]$ are immediately under the top-most "until" and some boolean combinators. We use the valid equalities from Lemma 8.3 to rewrite $f''[x]$ into an equivalent separated formula. There are a few special cases:

Case 1: If $\overline{\langle\langle\epsilon\rangle\rangle}x$ only occurs in the right-hand side of the "until", it is enough to put this right-hand side in disjunctive normal form, use (4), the distributivity law, to deal with disjunctions, and equalities (14) and (15), or, depending on k, (16) and (17), to obtain a separated $f'[x, \overline{\langle\langle\epsilon\rangle\rangle}x]$ with $f'[x, y]$ pure-future.

Case 2: If $\overline{\langle\langle\epsilon\rangle\rangle}x$ only occurs in the left-hand side of the "until", we use boolean manipulations to collect all these occurrences and put $f''[x]$ under the general form:

$$\Big((\overline{\langle\langle\epsilon\rangle\rangle}x \wedge \varphi) \vee (\neg\overline{\langle\langle\epsilon\rangle\rangle}x \wedge \varphi') \vee \beta\Big)\langle k\rangle\alpha$$

with pure-future φ, φ', β. Here we use (22) to obtain a separated $f'[x, \overline{\langle\langle\epsilon\rangle\rangle}x]$.

Case 3: If $\overline{\langle\langle\epsilon\rangle\rangle}x$ occurs in both sides if the "until", there are two subcases:

- if $k \neq \epsilon$, the distributivity law and equalities (14) and (15) are sufficient to eliminate right-hand side occurrences of $\overline{\langle\langle\epsilon\rangle\rangle}x$ so that we are back to Case 2.
- if $k = \epsilon$, this strategy does not work because (16) will bury α under two nested untils. That's why we developed the more complicated equalities (24) and (25) which, together with the distributivity law, will yield the answer we sought.

$\square$

A similar result is:

Lemma 8.5 *If $f[x]$ is a pure-future L_{BU} context and $b \in A$ is a visible label, then $f[\overline{\langle\langle b\rangle\rangle}x]$ is equivalent to some $f'[x, \overline{\langle\langle b\rangle\rangle}x, \overline{\langle\langle\epsilon\rangle\rangle}x]$ with $f'[x, y, z]$ pure-future and where y does not appear under the scope of "until" modalities.*

Proof By induction on the structure of $f[x]$. This follows the same steps we use for Lemma 8.4. Note that in $f'[x, y, z]$, z may appear under until modalities, so that $f'[x, \overline{\langle\langle b\rangle\rangle}x, \overline{\langle\langle\epsilon\rangle\rangle}x]$ is not necessarily separated. For this proof, this means that we may introduce new occurrences of $\overline{\langle\langle\epsilon\rangle\rangle}x$ (in pure-future contexts) and do not have to worry with any such occurrence that is already present.
Let's consider the induction step, assuming that $f[x]$ is an until-formula of the form $f_1[x]\langle k\rangle f_2[x]$. We look at $f[\overline{\langle\langle b\rangle\rangle}x]$. By ind. hyp., it is equivalent to some $f'_1[x, \overline{\langle\langle b\rangle\rangle}x, \overline{\langle\langle\epsilon\rangle\rangle}x]\langle k\rangle f'_2[x, \overline{\langle\langle b\rangle\rangle}x, \overline{\langle\langle\epsilon\rangle\rangle}x]$ where all occurrences of $\overline{\langle\langle b\rangle\rangle}x$ are immediately under the top-most until (and some boolean combinators).

Case 1 If $\overline{\langle\langle b\rangle\rangle}x$ only appears in the right-hand side of the "until", we use the distributivity law and equalities (18) to (21). Observe that (18) and (19) may introduce new occurrences of $\overline{\langle\langle\epsilon\rangle\rangle}x$.

Case 2 If $\overline{\langle\langle b\rangle\rangle}x$ only occurs in the left-hand sides of the "until", we use (23).

Case 3 In the general situation where $\overline{\langle\langle b\rangle\rangle}x$ occurs in both sides of the "until", we use (23) to extract the $\overline{\langle\langle b\rangle\rangle}x$'s from the left-hand side, and then (18) to (21) to extract them from the right-hand side.

□

Now we can merge Lemmas 8.4 and 8.5 into:

Lemma 8.6 *If $f[x]$ is a pure-future L_{BU} context, then $f[\overline{\langle\langle k\rangle\rangle}x]$ is equivalent to some separated $f'[x, \overline{\langle\langle k\rangle\rangle}x, \overline{\langle\langle \epsilon\rangle\rangle}x]$ with $f'[x,y,z]$ pure-future.*

Proof If $k = \epsilon$, this is directly Lemma 8.4. If $k = b \neq \epsilon$, we use Lemma 8.5 to get some $f'[x, \overline{\langle\langle b\rangle\rangle}x, \overline{\langle\langle \epsilon\rangle\rangle}x]$ where there only remains to extract all occurrences of $\overline{\langle\langle \epsilon\rangle\rangle}x$ from the "until" modalities, which is possible thanks to Lemma 8.4. □

We can build on this basic step:

Lemma 8.7 *If $f[x_1,\ldots,x_n]$ is a pure-future L_{BU} formula, then $f[\overline{\langle\langle k_1\rangle\rangle}x_1,\ldots,\overline{\langle\langle k_n\rangle\rangle}x_n]$ is equivalent to some separated $f'[x_1, \overline{\langle\langle k_1\rangle\rangle}x_1, \overline{\langle\langle \epsilon\rangle\rangle}x_1, \ldots, x_n, \overline{\langle\langle k_n\rangle\rangle}x_n, \overline{\langle\langle \epsilon\rangle\rangle}x_n]$ where $f'[x_1,y_1,z_1,\ldots,x_n,y_n,z_n]$ is pure-future.*

Proof By induction on n and using Lemma 8.6. □

Lemma 8.8 *If $f[x_1,\ldots,x_n]$ is a pure-future L_{BU} formula and if $\psi_1,\ldots,\psi_n$ are pure-past L_{BU} formulae, then $f[\psi_1,\ldots,\psi_n]$ is equivalent to a separated formula.*

Proof By induction on the maximum number of nested backward modalities in the ψ_i's, and using Lemma 8.7. □

Lemma 8.9 *If $f[x_1,\ldots,x_n]$ is a pure-future L_{BU} formula and if $\psi_1,\ldots,\psi_n$ are separated L_{BU} formulae, then $f[\psi_1,\ldots,\psi_n]$ is equivalent to a separated formula.*

Proof The ψ_i's may contain forward modalities in the scope of (nested) backward modalities. So that f is some $f[\psi_1[f_{1,1},\ldots,f_{1,k_1}],\ldots,\psi_n[f_{n,1},\ldots,f_{n,k_n}]]$ where the $f_{i,j}$'s are pure-future and where the $\psi_i[z_{i,1},\ldots,z_{i,k_i}]$'s are pure-past.
We apply Lemma 8.8 to $f[\psi_1[z_{1,1},\ldots,z_{1,k_1}],\ldots,\psi_n[z_{n,1},\ldots,z_{n,k_n}]]$ and get a separated $f'[z_{1,1},\ldots,z_{n,k_n}]$.
Then $f \equiv f'[f_{1,1},\ldots,f_{n,k_n}]$ which is separated. □

We can now prove Proposition 8.2: we show, by structural induction, that any f in L_{BU} is equivalent to a separated formula. The induction step is obvious in all cases except when f has the form $f_1\langle k\rangle f_2$, where we have to use Lemma 8.9.
The next step is simply:

Proposition 8.10 $L_{BU}^{sep} \preceq_i L_U$.

Proof Proceed as in the proof of Proposition 4.4, using $\overline{\langle\langle a\rangle\rangle}\varphi \equiv_i \perp$ and $\overline{\langle\langle \epsilon\rangle\rangle}\varphi \equiv_i \varphi$. $\square$

Now the proof of Theorem 8.1 is simply obtained as

$$L_B \subseteq L_{BU} \preceq_g L_{BU}^{sep} \preceq_i L_U$$

Incidentally, we can now generalize Theorem 7.2 with

Theorem 8.11 $L_{BU} \preceq_g L_B$.

Proof Consider $f \in L_{BU}$. Then f is equivalent to some $f' \in L_{BU}^{sep}$ (Proposition 8.2). f' is separated and thus has the form $\psi[\varphi_1, \ldots, \varphi_n]$ where $\psi[x_1, \ldots, x_n]$ is pure-past (and then in L_B) and the φ_i's are pure-future (and then in L_U). Theorems 7.1 and 7.2 imply that the φ_i's are (globally) equivalent to some φ_i''s in L_B. Finally, $f \equiv \psi[\varphi_1', \ldots, \varphi_n'] \in L_B$. $\square$

Figure 2 summarizes all the translation results we established in the branching bisimulation framework. Clearly no arrow (save those derived by transitivity) can be added because this would require translating (in the strong, "global equivalence", sense) a logic with backward modalities into a logic with only forward modalities.

9 Conclusion

In this article we proved that L_B, L_U and L_{sb} (three modal logics which have been proposed as characterizations of branching bisimulation) have the same expressivity. We gave effective translations between the three logics. The main technical difficulty lies in the fact that L_U and L_{sb} only have forward modalities while L_B has both forward and backward modalities.

An important question remains to be investigated: *what is the relative succinctness of the three logics?* All the translations we gave potentially lead to combinatorial explosion. This seems inescapable for the translations from L_B to L_{BU}^{sep} and from L_U to L_{sb}. Regarding the (straightforward) translation from L_{sb} to L_U, the combinatorial explosion disappears if we consider formulae as acyclic graphs rather than trees. Regarding the translation from L_{sb} to L_B, the same "graph versus tree" difficulty combines with the combinatorics of boolean conjunctive normal forms. Clearly, formally establishing non-polynomial lower bounds on relative succinctness would prove that none of L_B and L_U really subsumes the other. This would be a very strong argument in favor of using (say) L_{BU} as the natural modal logic for branching bisimulation.

More generally, translations between modal logics of reactive systems have not been subject to much investigation in the literature. This is partly due to the fact that few behavioral equivalences enjoy several distinct modal characterizations (in this regard, branching bisimulation was a welcome exception.) We believe many interesting translation problems can be investigated when modal logics with backward modalities are considered. For example, the logic L_P from [3] can be translated into a variant of HML_{bf} with modalities for *pomset* observations [20]. An interesting open problem regards HML with recursion, where we do not expect to develop translation algorithms based on rewrite rules. As an indication, let us mention that the linear-time μ-calculus with backward modalities can be translated (modulo $\equiv_i$) into the pure-future fragment [24] but the proof uses automata-theoretic techniques and it is not clear how to develop a translation operating on logic formulae.

10 Acknowledgments

It is a pleasure to acknowledge the very helpful suggestions, remarks, questions, ..., contributed by F. Vaandrager, R. van Glabbeek, G. Scollo and the anonymous referees at various stages of this work.

References

[1] S. D. Brookes and W. C. Rounds. Behavioural equivalence relations induced by programming logics. In *Proc. 10th ICALP, Barcelona, LNCS 154*, pages 97–108. Springer-Verlag, July 1983. Available as CMU Report CMU-CS-83-112.

[2] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.

[3] R. De Nicola and G. L. Ferrari. Observational logics and concurrency models. In *Proc. 10th Conf. Found. of Software Technology and Theor. Comp. Sci., Bangalore, India, LNCS 472*, pages 301–315. Springer-Verlag, December 1990.

[4] R. De Nicola, U. Montanari, and F. Vaandrager. Back and forth bisimulations. In *Proc. CONCUR'90, Amsterdam, LNCS 458*, pages 152–165. Springer-Verlag, August 1990.

[5] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation (extended abstract). In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, PA*, pages 118–129, June 1990.

[6] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation. Research Report SI-92/07, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", November 1992. To appear in J. ACM.

[7] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. Elsevier Science Publishers, 1990.

[8] D. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Proc. Temporal Logic in Specification, Altrincham, UK, LNCS 398*, pages 409–448. Springer-Verlag, April 1987.

[9] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. Principles of Programming Languages, Las Vegas, Nevada*, pages 163–173, January 1980.

[10] R. J. van Glabbeek. The linear time - branching time spectrum II: The semantics of sequential systems with silent moves. In *Proc. CONCUR'93, Hildesheim, Germany, LNCS 715*, pages 66–81. Springer-Verlag, August 1993.

[11] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in process algebra. In *Information Processing 89, Proc. IFIP 11th World Computer Congress, San Francisco*, pages 613–618. North-Holland, August 1989.

[12] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[13] M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, 67:23–52, 1985.

[14] M. Hillerström. Verification of CCS processes. M.Sc. Thesis, Aalborg University, 1987.

[15] H. Korver. Computing distinguishing formulas for branching bisimulation. In *Proc. CAV'91, Aalborg, LNCS 575*, pages 13–23. Springer-Verlag, 1991.

[16] F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. In *Proc. STACS'94, Caen, France, LNCS 775*, pages 47–58. Springer-Verlag, February 1994.

[17] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[18] R. Milner. A modal characterisation of observable machine-behaviour. In *Proc. CAAP'81, Genoa, LNCS 112*, pages 25–34. Springer-Verlag, March 1981.

[19] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[20] S. Pinchinat, F. Laroussinie, and Ph. Schnoebelen. Logical characterizations of truly concurrent bisimulation. Tech. Report 114, LIFIA-IMAG, Grenoble, March 1994.

[21] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion, LNCS 194*, pages 15–32. Springer-Verlag, July 1985.

[22] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Oxford Univ. Press, 1992.

[23] F. Vaandrager. Translating back and forth logic to HML with until operator. Unpublished note, 1992.

[24] M. Vardi. A temporal fixpoint calculus. In *Proc. 15th ACM Symp. Principles of Programming Languages, San Diego, CA*, pages 250–259, January 1988.

$$\begin{tikzcd} L_B \arrow[r, leftrightarrow, "g"] \arrow[d, shift left, "i"] & L_{BU} \arrow[r, leftrightarrow, "g"] & L_{BU}^{sep} \arrow[d, shift left, "i"] \\ L_0 \arrow[u, shift left, "g"] \arrow[rr, leftrightarrow, "g"] & & L_U \arrow[u, shift left, "g"] \end{tikzcd}$$

Figure 2: Translations between logics for branching bisimulation