Translations between modal logics of reactive systems*  

F. Laroussinie, S. Pinchinat, Ph. Schnoebelen*  

Lifia-Imag, 46 Av. Félix Viallet, F-38000 Grenoble, France  

Abstract  

We propose meaning-preserving translations between $L_B$, $L_U$ and $L_{ab}$ (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 $L_{ab}$ 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 (Gabbay, 1987). 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 $p \leftrightarrow q$, iff they satisfy the same HML formulae, written $p \equiv_{HML} q$, where  

$$p \equiv_{L} q \overset{\text{def}}{\iff} \forall f \in L (p \models f \iff q \models f).$$  

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  

*This article extends our earlier work presented in Proc. AMAST'93, Enschede, NL, June 1993.  
* Corresponding author. Email {fl,spinchi,phs} @lifia.imag.fr.
underlying bisimulation equivalence. Following the direction exemplified in [12],
many other behavioral equivalences have been 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 noncon-
tinuous properties of generalized transition systems ($J_T$ in [13]) to characterize
history-preserving bisimulation in causality-based models ($L_P$ in [3]) and to charac-
terize branching bisimulation by mimicking back-and-forth $\tau$-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, which 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 $\leftrightarrow_b$ for branching bisimulation, $p \equiv_L q$ iff $p \leftrightarrow_b q$ 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 tem-
poral 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 con-
structive, 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 ($\tau$'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 $\tau$'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) label. 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 establish 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 $(Q, \rightarrow)$ where $Q$ 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 $J_\tau$ defined in [13]).

**Definition 2.1.** $HML_{bf}$ formulae are built according to the following grammar:

$$HML_{bf} \varphi, \psi ::= \top \mid \neg \varphi \mid \varphi \land \psi \mid \langle a \rangle \varphi \mid \overline{\langle a \rangle} \varphi$$

where $a$ is any action from $A$.

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

2.2. Semantics

A modal logic with backward modalities states properties of a run $\pi = [q_0 \overset{a_1}{\rightarrow} q_1 \cdots \overset{a_n}{\rightarrow} q_n]$ of $S$. A run like $\pi$ 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 $\pi \overset{a_n}{\rightarrow} \pi'$ when run $\pi'$ is $\pi$ with a transition $q_n \overset{a_n}{\rightarrow} q_{n+1}$ 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. For a run $\pi$ of some LTS $S$ and an $HML_{bf}$ formula $f$, we define when $\pi \models_S f$ (reads "$\pi$ satisfies $f$") by induction on the structure of $f$:

\[
\begin{align*}
\pi \models T & \quad \text{always}, \\
\pi \models \neg f & \quad \text{iff } \pi \not\models f, \\
\pi \models f \land g & \quad \text{iff } \pi \models f \text{ and } \pi \models g, \\
\pi \models \langle a \rangle f & \quad \text{iff there is a } \pi' \text{ s.t. } \pi' \models f, \\
\pi \models \langle a \rangle^* f & \quad \text{iff there is a } \pi' \text{ s.t. } \pi' \models f.
\end{align*}
\]

(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 need 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

\[ q \models f \iff [q] \models f, \]

where $[q]$ 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 $p \models f \iff q \models f$ for all $f \in HML_{bf}$. De Nicola and Vaandrager [5] mention that $p \equiv_{HML_{bf}} q$ iff $p \equiv q$ because (strong) bisimulation coincides with (strong) back-and-forth bisimulation [4]. This entails

\[ p \equiv_{HML} q \iff p \equiv_{HML_{bf}} q. \]  

(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 $\langle a \rangle T$, which says that the last step was $a$-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 $q \models \langle a \rangle T$. Thus, in a certain sense, $\langle a \rangle T$ (an $HML_{bf}$ formula) can be correctly translated into $\bot$ (an $HML$ formula).

This requires some definitions.

Definition 3.1. Two formulae are globally equivalent, written $f \equiv f'$, iff $\pi \models f \iff \pi \models f'$ for all runs $\pi$ in all LTS's.

They are initially equivalent, written $f \equiv_i f'$, iff $q \models f \iff q \models f'$ for all states $q$ in all LTS's.
For example, we have $\langle a \rangle T \equiv_1 \bot$ but $\langle a \rangle T \not\equiv \bot$. Clearly, $f \equiv f'$ implies $f \equiv_1 f'$ 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 $\equiv_1$ 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 $L \leq_L L'$ (resp. $L \leq_{i} L'$) iff for any $f \in L$ there is an $f' \in L'$ with $f \equiv_1 f'$ (resp. $f \equiv_1 f'$).

Clearly, $L \leq_L L'$ implies $L \leq_{i} L'$. Also $L \leq L'$ implies $\equiv_L \subseteq \equiv_{L'}$. In both cases, the reverse implication is not true in general.

One trivial example is $HML \leq_{g} HML_{bf}$, which holds because $HML \subseteq HML_{bf}$. We now investigate the reverse direction.

4. From $HML_{bf}$ to $HML$.

**Theorem 4.1.** $HML_{bf} \leq_{i} HML$.

**Proof.** 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).

Here is the Separation Lemma for $HML_{bf}$.

**Proposition 4.2.**

$$HML_{bf} \leq_{g} HML_{bf}^{sep}.$$  \hspace{1cm} (2)

**Proof.** We show that any $f$ in $HML_{bf}$ is equivalent to a separated $f'$. The proof is done by structural induction on $f$. The cases when $f$ has the form $T$, $g_{1} \land g_{2}$, or $\neg g$ are obvious.

- $f = \langle a \rangle g$: $g$ can be separated (by induction hypothesis) into some $g'$. Then $f \equiv f' \overset{\text{def}}{=} \langle a \rangle g'$ is separated.
- $f = \langle a \rangle g$: $g$ can be separated (by induction hypothesis) into some $g'$. There are two subcases.
Assume $g'$ has the form $\langle b_1 \rangle \varphi_1 \land \cdots \land \langle b_n \rangle \varphi_n \land \neg \langle c_1 \rangle \varphi_1' \land \cdots \land \neg \langle c_m \rangle \varphi_m' \land \psi^+$ where $\psi^+$ is pure future. Write $c_1, \ldots, c_k$ for the $c_i$'s that are equal to $a$. Then

$$<a>g' = g'_\mathbf{def} = \begin{cases} \varphi_1 \land \cdots \land \varphi_n \land \neg \varphi_{i_1} \land \cdots \land \neg \varphi_{i_k} & \text{if } b_i = a \text{ for } i = 1, \ldots, n, \\ \bot & \text{otherwise.} \end{cases}$$

$f = g$ 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 $\langle b \rangle \varphi$, $\neg \langle b \rangle \varphi$, $\langle b \rangle \varphi$ or $\neg \langle b \rangle \varphi$. The $g_{i,j}$'s are separated. $f = g$ if $g' = \bigvee_i <a> (\bigwedge_j g_{i,j})$ and each $<a> (\bigwedge_j g_{i,j})$ falls in the previous subcase and can be separated. 

**Remark 4.3.** In a linear-time framework, Gabbay [8, 9] uses 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: $\langle a \rangle <b> T$ 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 the following proposition.

**Proposition 4.4.** $HML_{bf}^\mathbf{sep} \leq_i HML$.

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

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 $\tau$ for this invisible label and consider transition systems labeled over $A_\varepsilon = A \cup \{\tau\}$. We write $q \Rightarrow q'$ when there is a sequence $q \downarrow \cdots \downarrow q'$. That is, $\Rightarrow$ is the transitive and reflexive closure of $\downarrow$. In this setting, a very natural equivalence is branching bisimulation [11, 10]. De Nicola and Vaandrager [5] introduce $L_U$ 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. The formulae of $L_B$ are built according to the following grammar:

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

where $k$ is any label from $A_e \overset{\text{def}}{=} A \cup \{e\}$.

We use $[[k]] f$ and $[[[k]]] f$ as standard abbreviations.

The semantics of the new modalities is given by the following definition.

Definition 5.2.

\begin{align*}
\pi \models \langle a \rangle f & \text{ iff there is a } \pi \Rightarrow a \Rightarrow \pi' \text{ s.t. } \pi' \models f, \\
\pi \models \langle e \rangle f & \text{ iff there is a } \pi \Rightarrow a \Rightarrow \pi' \text{ s.t. } \pi' \models f, \\
\pi \models \langle a \rangle \langle e \rangle f & \text{ iff there is a } \pi' \Rightarrow a \Rightarrow \pi \text{ s.t. } \pi' \models f, \\
\pi \models \langle e \rangle \langle e \rangle f & \text{ iff there is a } \pi' \Rightarrow a \Rightarrow \pi \text{ s.t. } \pi' \models f.
\end{align*}

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 $f, \ldots, k \in A_e$,

(a) $\langle k \rangle (\bigvee_i f_i) \equiv \bigvee_i (\langle k \rangle f_i)$,
(b) $\langle k \rangle (\bigwedge_i f_i) \equiv \bigwedge_i (\langle k \rangle f_i)$,
(c) $\langle k \rangle \langle e \rangle f \equiv \langle e \rangle \langle k \rangle f \equiv \langle k \rangle f$,
(d) $\langle k \rangle \langle e \rangle f \equiv \langle e \rangle \langle k \rangle f \equiv \langle k \rangle f$,
(e) $\langle a \rangle \langle e \rangle f \equiv \langle a \rangle f$,
(f) $\langle a \rangle \langle e \rangle f \equiv \overline{\langle a \rangle} f$,
(g) $\langle e \rangle [[k]] f \equiv [[[k]]] f$.

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. The formulae of $L_U$ are built according to the following grammar:

$$L_U \models f, g := T \mid \neg f \mid f \wedge g \mid f \langle k \rangle g,$$

with $k \in A_e$. 
The semantics is given by the following definition.

**Definition 5.5.**

\[ \pi \models f(a)g \iff \exists n > 0, \pi = \pi_0 \xrightarrow{a} \pi_1 \xrightarrow{a} \cdots \xrightarrow{a} \pi_n \text{ s.t. } \pi_n \models g \text{ and } \pi_i \models f \text{ for } i < n, \]

\[ \pi \models f(e)g \iff \exists n \geq 0, \pi = \pi_0 \xrightarrow{e} \pi_1 \xrightarrow{e} \cdots \pi_n \text{ s.t. } \pi_n \models g \text{ and } \pi_i \models f \text{ for } i < n. \]

Then, the \( L_U \) formula \( f(a)g \) 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

\[ \langle k \rangle f \equiv \top \langle k \rangle (\top \langle e \rangle f), \]

while we do not see any way of expressing "until" as a combination of \( \langle . \rangle \) and \( \overline{\langle . \rangle} \) (and believe that no solution exists).

The only distributive property of "until" is

\[ f \langle k \rangle (g_1 \lor g_2) \equiv (f \langle k \rangle g_1) \lor (f \langle k \rangle g_2) \]

**5.3. \( L_{sb} \)**

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

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

\[ L_{sb} \models f, g := \top \mid \neg f \mid f \land g \mid f[k]g, \]

with \( k \in A_k \).

**Definition 5.7.**

\[ \pi \models f(a)g \iff \text{there is a } \pi \Rightarrow \pi' \xrightarrow{a} \pi'' \text{ s.t. } \pi'' \models g \text{ and } \pi' \models f, \]

\[ \pi \models f(e)g \iff \text{there is a } \pi \Rightarrow \pi' \text{ s.t. } \pi' \models f \text{ and } \]

\[ \pi' \models g \text{ or there is a } \pi' \xrightarrow{e} \pi'' \text{ with } \pi'' \models g. \]

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 runs between \( \pi \) and \( \pi' \). 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

\[
(f \lor f')[k]g \equiv f[k]g \lor f'[k]g
\]

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

\[
f[k]g \equiv \langle e \rangle(f \land f(k)g)
\]

entailing \( L_{sb} \preceq L_U \).

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:

\[
L_{BU}[f, g] = T | \neg f | f \land g | \langle k \rangle f | \langle k \rangle f | f(k)g | f[k]g,
\]

with \( k \in A \). (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 \( \equiv_{L_U} \), \( \equiv_L \) and \( \equiv_{L_{sb}} \) (and \( \equiv_{L_{sb,1}} \)) coincide, a natural question is whether any of the three logics can be translated into another. 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 the proof that \( L_U \preceq L_{sb} \preceq L_B \) and \( L_B \preceq_{1} L_U \). 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 \( \pi, \pi' \) in all LTS's, \( \pi \models f \) and \( \pi' \rightarrow \pi \) imply \( \pi' \models f \). It is a \( \Box \)-formula iff for all \( \pi, \pi' \) in all LTS's \( \pi \models f \) and \( \pi \rightarrow \pi' \) imply \( \pi' \models f \).
Thus, when a □-formula (resp. ◯-formula) holds of some π, it holds in all τ-successors (resp. τ-predecessors) of π. This is why for any □-formulae \( f^\circ \) and \( g^\circ \) and any ◯-formulae \( f^\diamond \) and \( g^\diamond \),

\[
(f^\circ \lor g^\circ) \langle k \rangle h \equiv (f^\circ \langle k \rangle h) \lor (g^\circ \langle k \rangle h),
\]

\[
(f^\diamond \lor g^\diamond) \langle k \rangle h \equiv (f^\diamond \langle k \rangle h) \lor (g^\diamond \langle k \rangle h).
\]

We write informally \( f \in □ \) (resp. \( f \in ◯ \)) when \( f \) is a ◯-formula (resp. a □-formula). A given formula may well be both a □- and a ◯-formula (witness \( T \) and \( \bot \)) or none.

The following properties are useful.

**Lemma 6.2.** For all \( f, g \in LBU \) and all \( k \in A_τ \),

(a) \( f \in ◯ \iff \neg f \in □ \),

(b) \( f \in □ \iff \neg f \in ◯ \),

(c) \( f, g \in ◯ \) implies \( f \land g, f \lor g \in ◯ \),

(d) \( f, g \in □ \) implies \( f \land g, f \lor g \in □ \),

(e) \( f \in ◯ \iff f \equiv \langle ε \rangle f \),

(f) \( f \in □ \iff f \equiv [[ε]] f \),

(g) \( \langle k \rangle f, [[k]] f \in ◯ \),

(h) [[k]] f, \( \langle k \rangle f \in □ \).

(i) \( f \{k\} g \in ◯ \).

**Proof.** (a)–(d) are clear from the definition, whereas (e) is left to the reader as a simple exercise. To prove (f), combine (b) and (e). To prove (g), combine (e) and Lemma 5.3(c) and (g). Use duality to prove (h). Finally, to prove (i), combine (6) and (g). □

Points (e) and (f) above may help understand our choice of terminology. With Lemma 6.2(i) above, we have the following important corollary.

**Corollary 6.3.** Any \( f \in Lsb \) is a boolean combination of ◯-formulae in \( Lsb \).

A similar result is true for \( LB \) also (witness Lemma 6.2(g) and (h)) but not for \( LU \) (witness \( \langle \tau \rangle (a) \langle T \rangle \langle b \rangle \langle T \rangle \)).

7. From \( LU \) to \( LB \)

All \( LU \) formulae (in fact, all \( LBU \) formulae, see Theorem 8.11) can be translated into \( LB \). In this section, we show how to go from \( LU \) to \( Lsb \) and then from \( Lsb \) to \( LB \).

**Theorem 7.1.** \( LU \preceq s Lsb \).
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 \( g(k)h \). Then, by induction hypothesis, \( g \) 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

\[
f \equiv \left( \bigvee_{t=1}^{n} (f_t^l \land g_t^l) \right) \langle k \rangle h',
\]

where, for \( i = 1, \ldots, n, f_t^l \in \varnothing \) and \( g_t^l \in \Box \). We now reason by induction on \( n \).

- First consider the simpler case where \( n = 1 \). We use

\[
(f^o \land g^o) \langle a \rangle h' \equiv g^o \land (f^o \{ a \} h'),
\]

and immediately obtain \( L_{sb} \) formulae.

- Now in the general case where \( n > 1 \), we use

\[
\left( \bigvee_{i=1}^{n} (f_t^l \land g_t^l) \right) \langle a \rangle h' \equiv \bigvee_{j=1}^{n} \left( g_j^o \land \left( f_j^l \{ a \} h' \lor f_j^l \{ g \} \left( \bigvee_{i=1}^{n} (f_t^l \land g_t^l) \langle a \rangle h' \right) \right) \right),
\]

\[
\left( \bigvee_{i=1}^{n} (f_t^l \land g_t^l) \right) \langle a \rangle h' \equiv h' \lor \bigvee_{j=1}^{n} \left( g_j^o \land f_j^l \{ g \} \left( \bigvee_{i=1}^{n} (f_t^l \land g_t^l) \langle a \rangle h' \right) \right),
\]

which can be translated by ind. hyp.

We let the reader check that (7)–(10) hold when \( f_t^l \in \varnothing \) and \( g_t^l \in \Box \) for all \( i \). As an indication, we can give the intuition behind (9): assume \( \pi \models (\bigvee_{i=1}^{s} (f_t^l \land g_t^l)) \langle a \rangle h \). Then there is a path \( \pi = \pi_0 \rightarrow \pi_1 \rightarrow \pi \) with \( \pi' \models h \) s.t. any \( \pi_s (0 \leq s \leq r) \) satisfies one of the \( f_t^l \land g_t^l \)'s. In particular, \( \pi \models f_j^l \land g_j^l \) for some \( j \) (and then \( \pi_{s} = g_j^l \) for \( s = 0, \ldots, r \)). Now there are two cases:

- either \( \pi_0, \ldots, \pi_r \) all satisfy \( f_j^l \land g_j^l \), and then \( \pi \models f_j^l \land (f_j^l \{ a \} h) \), as for (7),

- or there is a \( 0 < s \leq r \) s.t. \( \pi_s \not\models f_j^l \). We pick the smallest such \( s \). Then, because \( f_j^l \in \varnothing \), none of \( \pi_s, \pi_{s+1}, \ldots, \pi_r \) satisfies \( f_j^l \). Therefore they all satisfy \( (\bigvee_{i \neq j} (f_t^l \land g_t^l)) \langle a \rangle h \).

\[ \square \]

Theorem 7.2. \( L_{sb} \leq_{\varnothing} L_B \).

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 \( \varnothing \)-formulac. Then, thanks to (5) and Lemma 6.2, it is enough to only consider formulac of the general from \( (f^o \land g^o) \{ k \} h \), with \( f^o \in \varnothing \) and \( g^o \in \Box \). We
use
\[(f^o \land g^o) \{a\} h \equiv \langle\langle a \rangle\rangle (h \land [\langle a \rangle f^o \land \langle a \rangle g^o]), \tag{11}\]
\[(f^o \land g^o) \{a\} h \equiv \langle\langle a \rangle\rangle (f^o \land g^o \land \langle a \rangle (h \land [\langle a \rangle h \lor f^o])) \tag{12}\]
and there only remains to replace \(f^o\), \(g^o\) and \(h\) by their \(L_B\) equivalent. (Again, we let the reader check that (11) and (12) are valid whenever \(f \in \Phi\), \(g \in \Omega\).) \(\square\)

**Corollary 7.3.** \(L_U \leq s L_B\).

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 Theorem 8.1 as a corollary of Proposition 8.2, a Separation Lemma for \(L_{BU}\).

**Theorem 8.1.** \(L_B \leq_i L_U\).

**Proposition 8.2**

\[L_{BU} \leq_s L_{BU}^*\tag{13}\]

where \(L_{BU}^*\) 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 \(a, b \in A, k \in A_e\), we have
\[(\alpha \langle a \rangle \langle \overline{b} \rangle \varphi \land \beta) \equiv \alpha \langle a \rangle (\varphi \land \beta) \tag{14}\]
\[(\alpha \langle a \rangle \langle \neg \varphi \rangle \varphi \land \beta) \equiv \alpha \langle a \rangle (\neg \varphi \land \beta) \tag{15}\]
\[(\alpha \langle a \rangle \langle \varphi \rangle \varphi \land \beta) \equiv \langle \langle \varphi \rangle \varphi \land \alpha \langle a \rangle \beta \rangle \lor \alpha \langle a \rangle (\psi \land \alpha \langle e \rangle \beta) \tag{16}\]
\[(\alpha \langle a \rangle \langle \neg \varphi \rangle \varphi \land \beta) \equiv \neg \langle \langle \varphi \rangle \varphi \land (\alpha \land \neg \psi) \rangle \alpha \langle a \rangle (\beta \land \neg \psi) \tag{17}\]
\[(\alpha \langle a \rangle \langle \overline{b} \rangle \varphi \land \beta) \equiv \begin{cases} 1 & \text{if } a \neq b, \\ \langle \langle \overline{\varphi} \rangle \varphi \land \alpha \langle a \rangle \beta \rangle \lor \langle \langle \varphi \rangle \varphi \land \alpha \langle a \rangle \beta \rangle & \text{if } a = b. \end{cases} \tag{18}\]
\( \alpha \langle a \rangle (\neg \langle b \rangle \psi \land \beta) \equiv \begin{cases} \alpha \langle a \rangle \beta & \text{if } a \neq b, \\ \neg \langle e \rangle \psi \land (\alpha \land \neg \psi) \langle a \rangle \beta & \text{if } a = b. \end{cases} \)  

(19)

\( \alpha \langle e \rangle (\langle b \rangle \psi \land \beta) \equiv \langle b \rangle \psi \land \alpha \langle e \rangle \beta, \)

(20)

\( \alpha \langle e \rangle (\neg \langle b \rangle \psi \land \beta) \equiv \neg \langle b \rangle \psi \land \alpha \langle e \rangle, \)

(21)

\( ((\langle e \rangle \psi \land \varphi) \lor (\neg \langle e \rangle \psi \land \varphi') \lor \beta) \langle k \rangle \alpha \)

\( \equiv \beta \langle e \rangle \psi \land (\varphi \lor \beta) \langle k \rangle \alpha \lor \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi' \lor \beta)) \langle k \rangle \alpha \)

\( \lor \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi \lor \beta)) \langle e \rangle (\psi \land (\varphi \lor \beta) \langle k \rangle \alpha), \)

(22)

\( ((\langle b \rangle \psi \land \varphi) \lor (\neg \langle b \rangle \psi \land \varphi') \lor \beta) \langle k \rangle \alpha \)

\( \equiv (\langle b \rangle \psi \land (\varphi \lor \beta) \langle k \rangle \alpha) \lor (\neg \langle b \rangle \psi \land (\varphi' \lor \beta) \langle k \rangle \alpha), \)

(23)

\( ((\langle e \rangle \psi \land \varphi) \lor (\neg \langle e \rangle \psi \land \varphi') \lor \beta) \langle e \rangle (\langle e \rangle \psi \land \alpha) \)

\( \equiv \langle e \rangle \psi \land (\varphi \land \beta) \langle e \rangle \alpha \lor \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi' \lor \beta)) \langle e \rangle (\psi \land (\varphi \lor \beta) \langle e \rangle \alpha), \)

(24)

\( ((\langle e \rangle \psi \land \varphi) \lor (\neg \langle e \rangle \psi \land \varphi') \lor \beta) \langle e \rangle (\neg \langle e \rangle \psi \land \alpha) \)

\( \equiv \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi \lor \beta)) \langle e \rangle (\neg \psi \land \alpha). \)

(25)

**Proof.** All equivalences 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 \( \pi \models ((\langle e \rangle \psi \land \varphi) \lor (\neg \langle e \rangle \psi \land \varphi') \lor \beta) \langle k \rangle \alpha \). For simplicity, assume \( k \) is some visible \( a \). Then there is some \( \pi = \pi_0 \Downarrow \pi_1 \Downarrow \ldots \pi_{n-1} \Downarrow \pi_n \) s.t. \( \pi_n \models \alpha \) and \( \pi_i \models ((\langle e \rangle \psi \land \varphi) \lor (\neg \langle e \rangle \psi \land \varphi') \lor \beta) \) for all \( i < n \). We distinguish three cases (illustrated in Fig. 1).

**Case 1:** If \( \pi_0 \models \neg \langle e \rangle \psi \), then all \( \pi_i \)'s \( (0 \leq i < n) \) satisfy \( \neg \psi \) and then must satisfy \( \varphi \lor \beta \), so that \( \pi \models \langle e \rangle \psi \land (\varphi \lor \beta) \langle k \rangle \alpha \).

**Case 2:** Otherwise \( \pi_0 \models \neg \langle e \rangle \psi \). We have two subcases.

**Case 2.1:** Assume all \( \pi_i \)'s \( (0 \leq i < n) \) satisfy \( \neg \psi \). Then they all satisfy \( \neg \langle e \rangle \psi \) and then must all satisfy \( \varphi' \lor \beta \). So that \( \pi \models \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi' \lor \beta)) \langle k \rangle \alpha \).

**Case 2.2:** Otherwise the \( \pi_i \)'s satisfy \( \neg \langle e \rangle \psi \) for all \( i = 0, \ldots, m-1 \) where \( \pi_m \) (with \( 0 < m < n \)) is the first run in the sequence to satisfy \( \psi \). Then the remaining \( \pi_i \)'s \( (m \leq i < n) \) satisfy \( \langle e \rangle \psi \). In this case, \( \pi_i \) must satisfy \( \varphi' \lor \beta \) if \( 0 \leq i < m \), or \( \varphi \lor \beta \) if \( m \leq i < n \). Because \( \pi_m \models \psi \), we have \( \pi \models \neg \langle e \rangle \psi \land (\neg \psi \land (\varphi' \lor \beta)) \langle e \rangle \psi \land (\varphi \lor \beta) \langle k \rangle \alpha \).

Clearly, these three cases cover all possibilities. If now we assume \( k = e \), the same reasoning applies except that there is one more possibility: \( \pi \) may satisfy the left-hand
side of (22) by satisfying $\alpha$. In this case $\pi \models \zeta \langle k \rangle \alpha$ for any $\zeta$. As it also satisfies $\langle e \rangle \psi \lor \lnot \langle e \rangle \psi$, it must satisfy the disjunction $\langle e \rangle \psi \land (\varphi \lor \beta) \langle k \rangle \alpha \lor \lnot \langle e \rangle \psi \land (\lnot \psi \land (\varphi' \lor \beta)) \langle k \rangle \alpha$ and then the right-hand side of (22).

Now, it should be clear that if $\pi$ satisfies the right-hand side of (22), then we are necessarily in one of these three (or four, if $k=\varepsilon$) cases, so that

$$\pi \models (\langle e \rangle \psi \land \varphi) \lor (\lnot \langle e \rangle \psi \land \varphi') \lor (\varphi \lor \beta) \langle k \rangle \alpha.$$

We can now turn to the separation theorem for $L_{BU}$, that is we describe how equalities (14)–(25) allow to rewrite any $L_{BU}$ formula into an equivalent separated formula. Basically, (14)–(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 $\varphi$ in $f[x]$. We write $f[x_1, \ldots, x_n] = g[x_1, \ldots, x_n]$ when $f[\varphi_1, \ldots, \varphi_n] = g[\varphi_1, \ldots, \varphi_n]$ 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 \( f[\overline{\overline{e}}]x \) is equivalent to some separated \( f'[x, \overline{\overline{e}}]x \) 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 induction hypothesis, there are pure-future \( f'_1[x, y] \) and \( f'_2[x, y] \) s.t. \( f[\overline{\overline{e}}]x \) is equivalent to \( f'_1[x, \overline{\overline{e}}]x \langle k \rangle f'_2[x, \overline{\overline{e}}]x \), which we denote by \( f''[x] \). Because the \( f'_i[x, \overline{\overline{e}}]x \)'s are separated for \( i = 1, 2 \), all occurrences of \( \overline{\overline{e}} \) in \( f''[x] \) are immediately under the topmost “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 \( \overline{\overline{e}} \) 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 \( f'[x, \overline{\overline{e}}]x \) with \( f'[x, y] \) pure-future.

Case 2: If \( \overline{\overline{e}} \) 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

\[
(\overline{\overline{e}}x \land \varphi) \lor (\neg \overline{\overline{e}}x \land \varphi') \lor \beta \langle k \rangle \alpha,
\]

with pure-future \( \varphi, \varphi', \beta \). Here we use (22) to obtain a separated \( f'[x, \overline{\overline{e}}]x \).

Case 3: If \( \overline{\overline{e}} \) occurs in both sides of the “until”, there are two subcases:

* if \( k \neq e \), the distributivity law and equalities (14) and (15) are sufficient to eliminate right-hand side occurrences of \( \overline{\overline{e}}x \) so that we are back to Case 2.
* if \( k = e \), this strategy does not work because (16) will bury \( x \) under two nested unils.

That is 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 the following lemma.

Lemma 8.5. If \( f[x] \) is a pure-future \( L_{BU} \) context and \( b \in A \) is a visible label, then \( f[\langle b \rangle \overline{\overline{e}}]x \) is equivalent to some \( f'[x, \langle b \rangle \overline{\overline{e}}]x, \overline{\overline{e}}x \) 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 \( f'[x, \langle b \rangle \overline{\overline{e}}]x, \overline{\overline{e}}x \) is not necessarily separated. For this proof, this means that we may introduce new occurrences of \( \overline{\overline{e}}x \) (in pure-future contexts) and do not have to worry with any such occurrence that is already present.

Let us 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 \( f[\langle b \rangle \overline{\overline{e}}]x \). By induction hypothesis, it is equivalent
to some $f_1[x, \langle b \rangle x, \langle e \rangle x \langle k \rangle f_2[x, \langle b \rangle x, \langle e \rangle x\rangle]$ where all occurrences of \(\langle b \rangle x\) are immediately under the topmost until (and some boolean combinators).

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

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

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

Now we can merge Lemmas 8.4 and 8.5 into the following result.

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

**Proof.** If $k = e$, this is directly Lemma 8.4. If $k = b \neq e$, we use Lemma 8.5 to get some $f'[x, \langle b \rangle x, \langle e \rangle x]$ where there only remains to extract all occurrences of $\langle e \rangle x$ 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 $f[\langle k_1 \rangle x_1, \ldots, \langle k_n \rangle x_n]$ is equivalent to some separated $f'[x_1, \langle k_1 \rangle x_1, \langle e \rangle x_1, \ldots, x_n, \langle k_n \rangle x_n, \langle e \rangle x_n]$ 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 $\psi_i$’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 $\psi_i$’s may contain forward modalities in the scope of (nested) backward modalities. So that $f$ is some $f[\psi_1[f_1, 1, \ldots, f_{i, k_i}], \ldots, \psi_n[f_n, 1, \ldots, f_{n, k_n}]]$ where the $f_i$’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 \( f_i \langle k \rangle f_2 \), where we have to use Lemma 8.9.

The next step is simply the following proposition.

**Proposition 8.10.** \( L_{BU}^{sep} \leq_s L_U \).

**Proof.** Proceed as in the proof of Proposition 4.4, using \( \langle a \rangle \phi \equiv_i \bot \) and \( \ll \phi \gg \equiv_i \phi \). \( \square \)

Now the proof of Theorem 8.1 is simply obtained as

\[ L_B \subseteq L_{BU} \leq_s L_{BU}^{sep} \leq_i L_U. \]

Incidentally, we can now generalize Theorem 7.2 with the following theorem.

**Theorem 8.11.** \( L_{BU} \leq_s L_B \).

**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 \( \varphi_i \)'s are pure-future (and then in \( L_U \)). Theorems 7.1 and 7.2 imply that the \( \varphi_i \)'s are (globally) equivalent to some \( \varphi_i' \)'s in \( L_B \). Finally, \( f \equiv \psi[\varphi_1', \ldots, \varphi_n'] \in L_B \). \( \square \)

Fig. 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 \( L_{ub} \) (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 $L_{sp}$ 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 nonpolynomial 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 $HML_{bt}$ 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 $\mu$-calculus with backward modalities can be translated (modulo $\equiv$) 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.

Acknowledgment

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


