LTL Model Checking in Matching Logic

Jan Tušil
Masaryk University, Brno, Czech Republic
jan.tusil@mail.muni.cz

Xiaohong Chen
University of Illinois Urbana-Champaign, USA
xc3@illinois.edu

May 2, 2022

Abstract
Matching logic is a logic that serves as the foundation of the \( \mathbb{K} \) formal semantics framework. Through \( \mathbb{K} \), many programming languages have been given formal semantics in the form of a matching logic theory. The \( \mathbb{K} \) framework is then a best-effort implementation of matching logic reasoning. In this paper, we investigate the problem of LTL model checking of (potentially infinite) systems specified as matching logic theories. In particular, we construct a theory \( \Gamma_{MC} \) such that for any matching logic theory \( \Gamma_S \) (respecting certain syntactical constraints) representing a Kripke structure \( S \) and for any LTL formula \( \phi \), the structure \( S \) satisfies \( \phi \) iff \( \phi \) is valid in the joint theory \( \Gamma_S \cup \Gamma_{MC} \). This way, one can reason about the model checking problem using matching logic’s sound proof system. With this work, we hope to lay out the theoretical foundations for implementing LTL model checking in the \( \mathbb{K} \) framework, as well as to establish a connection between model checking and theorem proving.

1 Introduction
In an ideal world, formal verifiers would be sound and reusable. Every output result about a program, be it a refutation of a property or a proof of correctness, would be traceable back to the semantics of the given language, and reasoning techniques would be language-agnostic, directly applicable to all languages with no additional cost. Such is the vision pursued by the \( \mathbb{K} \) semantic framework (https://kframework.org) [9]. In \( \mathbb{K} \), developers define semantics of programming languages, and the framework then generates tools for manipulating and reasoning about programs in those languages, be it deductive verifiers or bounded model checkers. Real-world languages have been given semantics in \( \mathbb{K} \), including C [12] and Java [3].

In the heart of \( \mathbb{K} \) is matching logic [4,7,8,15,16]. Or, in the other direction, \( \mathbb{K} \) framework is a best-effort implementation of matching logic reasoning: a formal language definition in \( \mathbb{K} \) corresponds to a matching logic theory [4], and reasoning performed by \( \mathbb{K} \) is matching logic reasoning in that theory. It has been shown in [7] that matching logic subsumes one-path reachability logic (RL) [17].
a logic designed for language-agnostic, semantic-based reasoning about reach-
ability in deterministic languages. That allows $\mathcal{K}$ to perform RL-based reach-
ability analysis in a sound way: RL reasoning also counts as matching logic
reasoning.

The same paper [7] shows that matching logic can capture linear temporal
logic (LTL), and do so very elegantly, using only two axioms. Because of that,
one might be tempted to conclude by analogy that it is known how reason about
LTL properties of programs in matching logic; however, it is not the case. In
fact, more precisely, the result of [7], that $\models_{\text{LTL}} \varphi \iff \Gamma_{\text{LTL}} \vdash_{\text{ML}} \varphi$, shows only
that matching logic can capture LTL tautologies.

In this work, we show a similar but different axiomatization of LTL, which
exhibits a stronger property: it captures precisely LTL models (that is, traces).
Concretely, we define a theory $\Gamma_{\text{LTL}}$ and two model translation functions, a
function $\mathcal{M}$ from LTL models to matching logic $\Gamma_{\text{LTL}}$-models and a function $\mathcal{T}$
in the opposite direction, satisfying:

**Theorem (Model Equivalence).** We can translate an LTL formula $\varphi$ into a
matching logic formula $\psi$ such that: for any LTL trace $\tau$ and matching logic
$\Gamma_{\text{LTL}}$-model $M$,

$$\tau \models_{\text{LTL}} \varphi \iff \mathcal{M}(\tau) \vdash_{\text{ML}} \psi \quad \text{and} \quad \mathcal{T}(M) \models_{\text{LTL}} \varphi \iff M \vdash_{\text{ML}} \psi.$$ 

Moreover, we show that in matching logic, it is also possible to capture the
problem of LTL model checking of a Kripke structure. That is, we give a theory
$\Gamma_{\text{MC}}$ such that:

**Theorem (LTL Model Checking).** We can translate an LTL formula $\varphi$ into a
matching logic formula $\psi$ such that for any Kripke structure $S$ unambiguously
axiomatized as a theory $\Gamma^S$ in the open fragment of matching logic (Section 4.3),

$$S \models_{\text{LTL}} \varphi \iff \Gamma^S \cup \Gamma_{\text{MC}} \vdash_{\text{ML}} \psi.$$ 

That is, to verify (or refute) that a (potentially infinite) Kripke structure $S$
satisfies an LTL formula $\varphi$, one can verify (resp. refute) that the matching
logic formula $\psi$ holds in the theory that is the union of the axiomatization of $S$
with a generic “LTL model checking theory” $\Gamma_{\text{MC}}$. Since the verification can
be done using the existing sound proof system of matching logic, this way we
connect LTL model checking and theorem proving. Also, one can imagine that
the axiomatization $\Gamma^S$ of the Kripke structure $S$ can be given as a programming
language semantics, with the program under consideration as the initial state of
the structure. Therefore, we believe our work could serve as the theoretical basis
of a sound implementation of an LTL model checking tool in the $\mathcal{K}$ framework.

On the more technical side, we present a fragment of matching logic named
open fragment, together with techniques we named model extension and model
gluing, which allow us to extend a matching logic model with new elements, or
to concatenate two matching logic models together, such that the semantics of
the formulas from the open fragment is preserved. We used these techniques
to prove the main model checking theorem. Additionally, since they are more
generally useful, as they can be used to e.g. modularly prove consistency of a
union of two unrelated consistent theories, we present them in the main text as a
contribution for the matching logic community.

To recapitulate, our main contributions are:
• a matching logic theory $\Gamma^{\text{LTL}}$ capturing LTL models (Section 3);
• a matching logic theory $\Gamma^{\text{ltlmc}}$ capturing LTL model checking (Section 4);
and
• an idea of the open fragment supporting model extension and model gluing (Section 4.3).

All proof details can be found in the appendix.

2 Matching Logic Preliminaries

2.1 Matching Logic Syntax and Semantics

Matching logic \cite{5,7,8,15} is a seamless combination of first-order logic (FOL) and modal $\mu$-logic \cite{13} that makes no distinction between functions, predicates, or modal operators, but uses symbols to construct patterns, which capture static structures, dynamic properties, and logical constraints. It is parametric in two variable sets: (1) the set $\text{EV}$ of element variables, which are FOL-style variables that evaluate to individual elements; and (2) the set $\text{SV}$ of set variables that evaluate to sets. Given a matching logic signature $\Sigma$ as a set of symbols, the syntax of matching logic patterns \cite{5} is inductively defined:

$$\varphi ::= x | X | \sigma | \varphi_1 \varphi_2 | \bot | \varphi_1 \rightarrow \varphi_2 | \exists x . \varphi | \mu X . \varphi$$

where $\varphi$ is positive in $X$.

Here, $x$ ranges over element variables in $\text{EV}$, $X$ over set variables in $\text{SV}$, and $\sigma$ over symbols in $\Sigma$. The pattern $\varphi$ is positive in $X$, if no free occurrences of $X$ are nested an odd number of times on the left of an implication $\varphi_1 \rightarrow \varphi_2$. The construct $\varphi_1 \varphi_2$ is called application and is left-associative. Often, application has the form $\sigma \varphi$, where $\sigma \in \Sigma$ is a symbol. We use $\text{Pattern}(\Sigma)$ to denote the set of all patterns generated by the above grammar, and write $\varphi[\psi/x]$ to mean the result of substituting $\psi$ for $x$ in $\varphi$ while avoiding free variable capture. For notational simplicity, we define the syntactic sugar as usual:

$$\neg \varphi \equiv \varphi \rightarrow \bot \quad \forall x . \varphi \equiv \neg \exists x . \neg \varphi$$

$$\top \equiv \neg \bot \quad \varphi_1 \land \varphi_2 \equiv \neg (\neg \varphi_1 \lor \neg \varphi_2) \quad \nu X . \varphi \equiv \neg \mu X . \neg \varphi[\neg X/X]$$

The semantics of matching logic is based on pattern matching: patterns are interpreted as the sets of elements that match them. For example, $\varphi_1 \land \varphi_2$ is matched by elements that match both $\varphi_1$ and $\varphi_2$; $\varphi_1 \lor \varphi_2$ is matched by elements that match $\varphi_1$ or $\varphi_2$, etc. Element variable $x \in \text{EV}$ is matched by one single element, yielding the same semantics as FOL variables; $\exists x . \varphi$ builds abstraction, which is matched by all elements that can match $\varphi$ for some valuations of $x$; $\mu X . \varphi$ builds a least fixpoint pattern, which is matched by the elements in the smallest set $X$ such that $X = \varphi$ (note that $X$ may occur recursively in $\varphi$).

A matching logic model consists of (1) a nonempty carrier set $M$; (2) a function $\text{app}_M : M \times M \rightarrow \mathcal{P}(M)$ (where $\mathcal{P}(M)$ denotes the powerset of $M$) as the interpretation of application; and (3) a subset $\sigma_M \subseteq M$ as the interpretation of $\sigma \in \Sigma$. For notational simplicity, we use $M$ to denote both the carrier set and the whole model. We extend application from elements to sets: $\text{appext}_M(A, B) = \bigcup_{a \in A, b \in B} \text{app}_M(a, b)$ for $A, B \subseteq M$. The semantics of patterns
is given using a valuation $\rho$: $(Ev \rightarrow M) \cup (SV \rightarrow P(M))$. The interpretation of a pattern $\varphi$, denoted as $|\varphi|_{M,\rho}$ or $|\varphi|_{\rho}$, is a set of model elements defined inductively:

- $|x|_{\rho} = \{\rho(x)\}$ for $x \in Ev$
- $|X|_{\rho} = \rho(X)$ for $X \in Sv$
- $|\bot|_{\rho} = \emptyset$
- $|\varphi_1 \varphi_2|_{\rho} = M \ \setminus \ (|\varphi_1|_{\rho} \ \setminus \ |\varphi_2|_{\rho})$
- $|\mu X . \varphi|_{\rho} = \text{lfp}(A \mapsto |\varphi|_{\rho}(A/X))$

where "\-" denotes set difference; $\rho[a/x]$ (resp. $\rho[X]$) denotes valuation update, which is the valuation $\rho'$ such that $\rho'(x) = a$ (resp. $\rho'(X) = A$) and agrees with $\rho$ on all other variables; $\text{lfp}$ denotes the true least fixpoint, since $A \mapsto |\varphi|_{M,\rho(A/X)}$ is provably monotone (Lemma 25 in the appendix) and thus have a unique least fixpoint by the Kneser-Tarski fixpoint theorem [22].

**Definition 1.** A model $M$ satisfies a formula $\varphi$, written $M \models \varphi$, iff the formula is interpreted as the whole set $M$ in all valuations; that is, iff $|\varphi|_{\rho} = M$ for all $\rho$. A theory $\Gamma$, which is a set of matching logic patterns called axioms, is satisfied in a model $M$, written $M \models \Gamma$, iff $M \models \varphi$ for all $\varphi \in \Gamma$. We define $\Gamma \models \varphi$ iff $M \models \varphi$ for all $M \models \Gamma$, and let $\text{Mod}_{ML}(\Gamma) = \{M \ | \ M \models \Gamma\}$ be the class of all models of the theory $\Gamma$. For any ML pattern $\varphi$, let $FV(\varphi)$ be the set of all free variables of $\varphi$. When $FV(\varphi) = \emptyset$, the interpretation of $\varphi$ does not depend on the valuation, so we sometimes use the notation $|\varphi|_{M}$ to denote the unique interpretation of $\varphi$ in the model $M$.

### 2.2 Equality, Inclusion, Membership

We show how to define equality $\varphi_1 = \varphi_2$, set inclusion $\varphi_1 \subseteq \varphi_2$, and membership $x \in \varphi$ in an axiomatically way in matching logic as a logical theory. Specifically, we define the equality $\varphi_1 = \varphi_2$ (similar for inclusion and membership) as a matching logic pattern such that it holds (i.e., evaluates to $\top$), iff $\varphi_1$ evaluates to the same set as $\varphi_2$, and it fails (e.g., evaluates to $\bot$) otherwise.

<table>
<thead>
<tr>
<th>Spec DEFINEDNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symbols:</strong></td>
</tr>
<tr>
<td>$def$</td>
</tr>
<tr>
<td><strong>Notations:</strong></td>
</tr>
<tr>
<td>$[\varphi] \equiv def \varphi$</td>
</tr>
<tr>
<td>$\varphi \equiv x \in \varphi \equiv [x \land \varphi]$</td>
</tr>
<tr>
<td>$\varphi \equiv \neg [\neg \varphi]$</td>
</tr>
<tr>
<td>$\varphi_1 = \varphi_2 \equiv [\varphi_1 \leftrightarrow \varphi_2]$</td>
</tr>
<tr>
<td>$x \notin \varphi \equiv \neg (x \in \varphi)$</td>
</tr>
<tr>
<td>$\varphi_1 \neq \varphi_2 \equiv \neg (\varphi_1 = \varphi_2)$</td>
</tr>
<tr>
<td>$\varphi_1 \not\subseteq \varphi_2 \equiv \neg (\varphi_1 \subseteq \varphi_2)$</td>
</tr>
</tbody>
</table>

Spec. 1: Definedness and related notions

We define equality, inclusion, and membership as shown in Spec. 1. The specification defines a matching logic signature $\Sigma_{\text{DEFINEDNESS}}$ containing one
Spec SORTS Import: DEFINEDNESS Symbols: \textit{inh}

Notations:

\begin{align*}
\llbracket s \rrbracket & \equiv \text{inh } s \\
\neg \varphi & \equiv (\neg \varphi) \land \llbracket s \rrbracket \\
\forall x : s . \varphi & \equiv \forall x . x \in \llbracket s \rrbracket \land \varphi \\
\exists x : s . \varphi & \equiv \exists x . x \in \llbracket s \rrbracket \land \varphi \\
f : s_1 \times \cdots \times s_n \rightarrow s & \equiv \forall x_1 : s_1 \ldots \forall x_n : s_n . \exists y : s . f x_1 \ldots x_n = y \\
f : s_1 \times \cdots \times s_n \rightarrow s & \equiv \forall x_1 : s_1 \ldots \forall x_n : s_n . \exists y : s . f x_1 \ldots x_n \subseteq y
\end{align*}

Spec. 2: Sorts and sorted quantification

symbol, called “definedness”, a theory $\Gamma^\text{DEFINEDNESS}$ containing one axiom, called (DEFINEDNESS), and a few notations, which allow us to write, e.g., $\llbracket \varphi \rrbracket$ instead of $\text{def } \varphi$. The axiom (DEFINEDNESS) enforces that in all models, $\llbracket \varphi \rrbracket_{\rho} = M \iff \llbracket \neg \varphi \rrbracket_{\rho} = \emptyset$. (It also holds that $\llbracket \varphi \rrbracket_{\rho} = \emptyset$ otherwise, even without the axiom.) We use the name “definedness” for the symbol and the axiom, since the pattern $\llbracket \varphi \rrbracket$ means that $\varphi$ is defined; i.e., matched by at least by one element. With the definedness symbol and axiom, we can define equality, membership, and set inclusion, as syntactic sugar, using another notation, $\llbracket \varphi \rrbracket$, called “totality”. Intuitively, $\llbracket \varphi \rrbracket$ states that $\varphi$ is matched by all elements - we say that $\varphi$ is “total”. Then, $\varphi_1 = \varphi_2$ states that $\varphi_1$ and $\varphi_2$ are matched by the same elements, etc.

Unlike FOL, where formulas evaluate to either true or false, matching logic patterns can evaluate to any sets. However, some patterns, such as $\varphi_1 = \varphi_2$, can evaluate only to the empty set or the total set. We call such patterns predicate patterns. Intuitively, the purpose of predicate patterns is to make a statement. If a predicate pattern evaluates to the empty set, it means that the statement is false, while if it evaluates to the total set, the statement is true. So for example, the pattern $\llbracket \neg \varphi \rrbracket$ make the statement that $\varphi$ evaluates to a nonempty set (i.e., matches something); the pattern $x \in \varphi$ says that $\varphi$ matches $x$. Predicate patterns are closed over boolean connectives and quantification, and they correspond the first-order logic formulas. We often use predicate patterns to specify a condition on elements being matched. For example, the pattern $x \land (x \neq y)$ can match any element distinct from $y$.

2.3 Sorts

In Section\ref{sec:sorts} we need to represent LTL traces and their suffixes. For this purpose, we use \textit{sorts}. Although matching logic has no builtin support for sorts or many-sorted functions, it can represent them by symbols defined with proper axioms. Specifically, for every sort $s$, we define a corresponding symbol also denoted $s$ to represent its name, and define a symbol $\llbracket \rrbracket$, called \textit{inhabitant}, with the intuition that $\llbracket s \rrbracket$ is matched by all elements of sort $s$. Then, we can specify properties about sorts by patterns.

We define \textit{sorted quantification} that requires $x$ to have sort $s$ in Spec.\ref{spec:2} Here we also define \textit{sorted definedness} and \textit{sorted totality}. Intuitively, sorted
definedness $\lceil \phi \rceil_s$ says that the pattern $\phi$ matches at least one element of the sort $s$, while sorted totality $\lfloor \phi \rfloor_s$ says that $\phi$ matches all elements of the sort $s$. The two are dual, in the sense that $\models \lfloor \phi \rfloor_s = \neg \lceil \neg \phi \rceil_s$. Additionally, we define sorted functions and sorted partial functions as expected.

3 Defining LTL in Matching Logic

In this section we review linear temporal logic (LTL) \cite{14}, define it as a matching logic specification $LTL$, define translations from LTL to matching logic models and back, and show that our definitions are well-behaved by proving model equivalence theorems. We also compare our axiomatization of LTL with those of \cite{7}, which is simpler and does not capture models. In this section we focus on conveying the intuition behind our constructions; we refer an interested reader to the Appendix.

3.1 LTL Syntax and Semantics

The syntax of LTL is parametric in a countable set $AP$ of atomic propositions denoted as $a$; from now on, let $AP$ be fixed. The set $\Phi_{LTL}$ of LTL formulas is defined by the following grammar:

$$\psi ::= a \mid \neg \psi \mid \psi \land \psi \mid o\psi \mid \psi U \psi$$

Intuitively, $o\psi$ holds in a state of a path iff $\psi$ holds in the next state of the path (path is an infinite sequence of states); $\psi_1 U \psi_2$ holds on a state of a path iff $\psi_2$ holds in some future state of that path and all states until that point satisfy $\psi_1$.

An infinite trace $\tau \in (P(AP))^\omega$, or simply trace, is an infinite sequence of subsets of atomic propositions. We write $\tau[i]$ to mean the $i$th (starting from 1) element of trace $\tau$ for $i \geq 1$. We use $\tau[i..]$ to denote the suffix trace $\tau[i], \tau[i + 1], \ldots$.

LTL models are infinite traces. We let $\text{Mod}_{LTL} = (P(AP))^\omega$ denote the set of all LTL models. The semantics of LTL formula is defined w.r.t. a trace $\tau$ as the relation $\models_{LTL} \subseteq \text{Mod}_{LTL} \times \mathbb{N}_{\geq 1} \times \Phi_{LTL}$ inductively defined as follows:

- $\tau, i \models_{LTL} a$ iff $a \in \tau[i]$;
- $\tau, i \models_{LTL} \neg \psi$ iff $\tau, i \not\models_{LTL} \psi$;
- $\tau, i \models_{LTL} \psi_1 \land \psi_2$ iff $\tau, i \models_{LTL} \psi_1$ and $\tau, i \models_{LTL} \psi_2$;
- $\tau, i \models_{LTL} o\psi$ iff $\tau, i + 1 \models_{LTL} \psi$;
- $\tau, i \models_{LTL} \psi_1 U \psi_2$ iff there exists $j \geq i$ such that $\tau, j \models_{LTL} \psi_2$ and for all $i \leq k < j$ we have $\tau, k \models_{LTL} \psi_1$.

We write $\tau \models_{LTL} \psi$ if $\tau, 1 \models_{LTL} \psi$ and $\models_{LTL} \psi$ if $\tau \models_{LTL} \psi$ for all $\tau \in (P(AP))^\omega$.

3.2 Capturing LTL in Matching Logic

We define a matching logic specification $LTL$ that captures precisely LTL models, shown in Spec. 3; the specification defines a signature $\Sigma_{LTL}$ and a $\Sigma_{LTL}$-theory $\Gamma_{LTL}$. We explain the specification as follows. Models of LTL formulas are traces; therefore, we introduce a sort $Trace$ whose only inhabitant is a trace.
from $(\mathcal{P}(\text{AP}))^\omega$. To determine whether a trace satisfies a LTL formula, one needs to consider the suffixes of the trace. A trace suffix is, intuitively, a trace paired with an offset; We represent trace suffixes using the sort $\text{TraceSuffix}$, and identify (full) traces with trace suffixes paired with 1. The intuition is that a $\Sigma^{\text{LTL}}$-pattern representing an LTL formula matches exactly those trace suffixes that satisfy the formula. We also include a symbol $\text{tr}_a$ for every atomic proposition; the symbol is intended to match any trace suffix $\tau, i$ whose first state $\tau[i]$ satisfies the atomic proposition. The behavior of this symbol is axiomatized by the axiom schema $(\text{AtomicProp})$, where the metavariable $a$ ranges over all atomic propositions.

<table>
<thead>
<tr>
<th>Spec LTL Import: SORTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols:</td>
</tr>
<tr>
<td>$\text{Trace}$, $\text{TraceSuffix}$, $\circ$, $\bar{o}$, $\text{tr}_a$ (for every $a \in \text{AP}$)</td>
</tr>
<tr>
<td>Notations:</td>
</tr>
<tr>
<td>$\top_T \equiv [\text{Trace}]$</td>
</tr>
<tr>
<td>$\top_{TS} \equiv [\text{TraceSuffix}]$</td>
</tr>
<tr>
<td>$\neg^{\text{LTL}} \varphi \equiv \top_{TS} \wedge \neg \varphi$</td>
</tr>
<tr>
<td>$\varphi_1 U \varphi_2 \equiv \mu X . \varphi_2 \vee (\varphi_1 \wedge \circ X)$</td>
</tr>
<tr>
<td>Axioms:</td>
</tr>
<tr>
<td>$(\text{Prev}) \bar{o} x = \exists y : \text{TraceSuffix} . y \wedge (x \in \circ y))$</td>
</tr>
<tr>
<td>$(\text{Trace}) \exists x : \text{Trace} . \top_{T} = x$</td>
</tr>
<tr>
<td>$(\text{TraceSuffix}) \top_{TS} = \mu X . \top_{T} \vee \bar{o} X$</td>
</tr>
<tr>
<td>$(\text{Init}) \circ [\text{Trace}] = \bot$</td>
</tr>
<tr>
<td>$(\text{PrevFun}) \circ : \text{TraceSuffix} \rightarrow \text{TraceSuffix}$</td>
</tr>
<tr>
<td>$(\text{NextPFun}) \circ : \text{TraceSuffix} \mapsto \text{TraceSuffix}$</td>
</tr>
<tr>
<td>$(\text{AtomicProp})_a \text{tr}<em>a \subseteq \top</em>{TS}$</td>
</tr>
</tbody>
</table>

Spec. 3: LTL as a matching logic specification

Now we explain the main operators/notations that we define; namely, $\neg^{\text{LTL}}$, $\circ$, $\bar{o}$, and $U$.

1. The notation $\neg^{\text{LTL}}$ represents negation of an LTL formula. We cannot use the built-in matching logic negation for the purpose of negating an LTL formula, since the carrier set may contain elements other than trace suffixes, and these would be included in the result of negation. For example, in some model, the pattern $\neg\text{tr}_a$ matches the definedness symbol.

2. The symbol $\circ$ represents the LTL “next” operator. The intended meaning of this operator is the following: the pattern $\circ \varphi$ matches (“holds in”)
the trace suffix $\tau, i$ iff $\varphi$ matches (“holds in”) $\tau, i + 1$. See the following illustration:

\[
\begin{array}{cccc}
\tau, i & \tau, i + 1 & \tau, i + 2 & // \text{trace suffixes} \\
\circ \circ \varphi & \circ \varphi & \varphi & // \text{patterns}
\end{array}
\]

Therefore, perhaps counter-intuitively, we can view the operator $\circ$ (“next”) as a function in the opposite direction: if it takes as an argument $\tau, i + 1$, it returns $\tau, i$ - intuitively, it extends the trace suffix given as the argument with the “previous” state $\tau[i]$. Consequently, one can obtain the (full) trace $\tau, 1$ of a trace suffix $\tau, i$ by repeatedly applying $\circ$.

3. The operator $\bar{\circ}$ (“previous”) is a symbol defined by the axiom (Prev) to be the inverse of $\circ$; i.e., given a trace suffix $\tau, i$ as the argument, $\bar{\circ}$ returns $\tau, i + 1$. The axiom intuitively says that $\bar{\circ} z$ returns the set of all traces for which $\circ$ yields $z$. In matching logic, there exists an idiomatic way to express the set of all elements that satisfy a given property (i.e., set comprehension): the scheme $\exists x. x \land P(x)$, where $P(x)$ is a predicate. Recall that in matching logic, existential quantifier is interpreted as a union; only elements satisfying given predicate contribute to the union, because the other elements get filtered out by intersection with the failing predicate, i.e., the empty set.

4. The operator $U$ (“until”) is defined as a notation over $\circ$. Recall that $\varphi_1 U \varphi_2$ intuitively means that “eventually, $\varphi_2$ holds, and until that point, $\varphi_1$ holds”. This operator has a recursive nature, since the pattern $\varphi_1 U \varphi_2$ can be expand to semantically equivalent $\varphi_2 \lor (\varphi_1 \land (\varphi_1 U \varphi_2))$. The pattern $\mu X. \varphi_2 \lor (\varphi_2 \land \circ X)$ implements $U$ using this expansion.

With the symbols and notations in place, LTL formulas become matching logic patterns, assuming that negation $\neg \psi$ is mapped to sorted negation $\neg_{\text{LTL}} \psi$.

For any $\Gamma_{\text{LTL}}$-model $M$, by axiom (Trace), $\models \tau | M$ is a singleton set; we denote its unique element as $M_{\text{Trace}}$. We also write $M_{\text{TraceSuffix}}$ to mean $\models \tau S | M$. Intuitively, $M_{\text{Trace}}$ is the trace represented by the model $M$, while $M_{\text{TraceSuffix}}$ represents all the suffixes of the trace $M_{\text{Trace}}$. Now we have the following property:

**Lemma 2.** For a $\Gamma_{\text{LTL}}$-model $M$, a LTL formula $\varphi$, is matched only by trace suffixes; i.e., $|\varphi|_M \subseteq M_{\text{TraceSuffix}}$.

### 3.3 From LTL Models to Matching Logic $\Gamma_{\text{LTL}}$-Models

In this section we prove the first model equivalence theorem, which says that an LTL model satisfies a formula if and only if its translation to matching logic satisfies the translated formula. We defer the concrete construction of the model translation function $M : \text{Mod}_{\text{LTL}} \rightarrow \text{Mod}_{\text{ML}}(\Gamma_{\text{LTL}})$ to appendix (see Definition 29). Here, we only need to know that the carrier set of the constructed model contains positive natural numbers: $M(\tau)_{\text{TraceSuffix}} = \mathbb{N}_{\geq 1}$ for any LTL model (i.e., a trace) $\tau$, and that $\circ$ is interpreted as a partial
function decrementing the given number and \( \bar{o} \) as a total function incrementing the given number.

We said earlier that the intuition behind the sort \( \text{TraceSuffix} \) is that a \( \Sigma\text{LTL} \) pattern representing an LTL formula matches exactly those trace suffixes that satisfy the formula. The following lemma makes the intuition precise, thus connecting the semantics of LTL and matching logic.

**Lemma 3.** \( \tau, i \models_{\text{LTL}} \varphi \iff i \in |\varphi|_{M(\tau)} \) for any \( \tau \in \text{Mod}_{\text{LTL}}, i \geq 1 \), and a formula \( \varphi \).

Now we would like to prove Theorem 4, which connects LTL validity with matching logic validity. However, we must be careful when formulating such proposition, because the pattern \( \varphi \) is matched only by trace suffixes (Lemma 2), while a matching logic pattern is valid if it matches all elements. Since we only want the pattern \( \varphi \) to match the (full) trace, we state:

**Theorem 4 (Model Equivalence A).** For any LTL model \( \tau \in \text{Mod}_{\text{LTL}} \) and a formula \( \varphi \),

\[ \tau \models_{\text{LTL}} \varphi \iff M(\tau) \models [\varphi]_{\text{Trace}} \]

Intuitively, this theorem says that \( \tau \) satisfies \( \varphi \) in LTL iff the translated formula matches the full trace component of the model \( M(\tau) \); i.e., the element \( M_{\text{Trace}} \).

### 3.4 From Matching Logic \( \Gamma_{\text{LTL}} \)-Models to LTL Models

In this section we prove Theorem 5, which says, intuitively, that any matching logic model of \( \Gamma_{\text{LTL}} \) can be transformed to an LTL model that satisfies the same set of LTL formulas. Let us assume that we have a matching logic model \( M \models \Gamma_{\text{LTL}} \). Our goal is to construct a model \( T(M) \) yielding the same semantics as \( M \). Recall that an LTL model is an infinite trace over a set of atomic proposition. Therefore, we need to specify the atomic propositions and the infinite trace. Intuitively, we build the following infinite trace \( \tau[1], \tau[2], \tau[3], \ldots \), where \( \tau[i] \) is the set of atomic propositions that are satisfied in the trace suffix \( \tau, i \) extracted from \( M \). First, we need a way to address those elements of a \( \Gamma_{\text{LTL}} \)-model that represent trace suffixes.

**Definition 5.** For any \( \Gamma_{\text{LTL}} \)-model \( M \), we define the function \( M_\bar{o} : M \rightarrow M \) defined by \( M_\bar{o}(m) = m' \), where \( m' \) is the unique element satisfying \( \{m'\} = \text{appext}_M(\bar{o}_M, \{m\}) \), and the function \( [\_]_{\text{LTL}} : \mathbb{N} \geq 1 \rightarrow [\text{TraceSuffix}]_M \) by

\[
[n]_{M}^{\text{LTL}} = \begin{cases} 
M_{\text{Trace}} & \text{if } n = 1 \\
M_\bar{o}(\lceil n-1 \rceil_{M}^{\text{LTL}}) & \text{if } n > 1
\end{cases}
\]

Intuitively, \( [\_]_{M}^{\text{LTL}} \) represents the suffix \( \tau, i \) of the trace \( \tau \) represented by the model \( M \). With this construction, we can now extract an LTL model out of \( M \).

**Definition 6.** Let \( T : \text{Mod}_{\text{ML}}(\Gamma_{\text{LTL}}) \rightarrow \text{Mod}_{\text{LTL}} \) be the model translation function from matching logic to LTL, defined by \( T(M)(i) = \{a \in \text{AP} \mid [i]_{M}^{\text{LTL}} \in a_M\} \) for every \( M \in \text{Mod}_{\text{ML}}(\Gamma_{\text{LTL}}) \) and \( i \in \mathbb{N} \geq 1 \).
The important question is: how does satisfaction in $M$ relate to satisfaction in $T(M)$? The key insight is that from Section 3.3 we already know how does satisfaction in $T(M)$ relate to matching in $M(T(M))$, by instantiating Lemma 3 with $\tau = T(M)$. If we could tie matching in $M(T(M))$ to matching in $M$, we would get the relationship between matching in $M$ and satisfaction in $T(M)$ by transitivity.

We notice that the set of trace suffixes $M_{TraceSuffix}$ of a $\Gamma^{LTL}$-model $M$ has a structure that is similar to the set of natural numbers: the axiom (TraceSuffix) defining the inhabitant set of the sort $TraceSuffix$ resembles the axiom used for defining natural numbers, with $\top_T$ playing the role of $0$ and $\bar{\circ}$ playing the role of the successor function. We also have a function $[\|]\_M^{LTL}$ from positive natural numbers to the members of the inhabitant set of the sort $TraceSuffix$. We now make the relationship precise: we define a function $dist_M$ going in the other direction and show that the two are inversions of each other. Then we show that $dist_M$ and $[\|]\_M^{LTL}$ preserve matching of LTL formulas, and are therefore exactly the ties between $M$ and $M(T(M))$ that we need.

Axioms (Trace) and (TraceSuffix) together with the injectivity of $\bar{\circ}$ (see Lemma 28 in the appendix) ensure that every element $m \in M_{TraceSuffix}$ can be uniquely constructed by repeatedly applying $\bar{\circ}$ to $M_{Trace}$ - intuitively, every trace suffix can be uniquely constructed by repeatedly incrementing the offset of the trace represented by $M$. We define the function $dist_M: M_{TraceSuffix} \rightarrow \mathbb{N}_{\geq 1}$ by $dist_M(m) = n + 1$, where $n$ is the number of times $\bar{\circ}$ needs to be applied to get from $M_{Trace}$ to $m$. In other words, $dist_M(m)$ is the $\bar{\circ}$-distance of $m$ from $M_{Trace}$. We call the (unique) sequence of trace suffixes leading from $M_{Trace}$ to $m$ the initial sequence of $m$.

The main component of matching logic models built from LTL models is $\top_{TS,M(\tau)}$, which is the set of positive natural numbers. The following lemma relates $M_{TraceSuffix}$ of an arbitrary $\Gamma^{ML}$-model to $M(T(M))_{TraceSuffix}$.

**Lemma 7.** Functions $[\|]\_M^{LTL}$ and $dist_M$ are inversions of each other, and therefore are bijections between $M_{TraceSuffix}$ and $M(T(M))_{TraceSuffix}$.

The relationship between $M$ and $M(T(M))$ is stronger: the two models are indistinguishable by LTL formulas. We can take a LTL formula, evaluate it in the model $M$, and then translate the matched elements (“trace suffixes”) to elements of $M(T(M))$ - and we get the same result as if we evaluated the formula directly in $M(T(M))$ (see Figure 1).

**Lemma 8.** For every $M \in \text{Mod}_{ML}(\Gamma^{LTL})$ and $\varphi \in \Phi_{LTL}$, $\varphi|_{M(T(M))} = dist_M(\varphi|_{M})$.

And since $dist_M$ is a bijection, we can do the same in the opposite direction. Therefore, we are able to prove the following counterpart of Lemma 3.

**Lemma 9.** For all $M \in \text{Mod}_{ML}(\Gamma^{LTL})$ and $\varphi \in \Phi_{LTL}$,

$$T(M), i \models_{LTL} \varphi \iff [i]^M \in \varphi|_{M}.$$  

The proof goes as suggested above: we relate the satisfaction in $T(M)$ to matching in $M(T(M))$ using Lemma 3 and then use Lemma 8 and Lemma 7 to bring the reasoning back to $M$. With Lemma 7 it is now easy to prove the second model equivalence theorem.
Theorem 10 (Model Equivalence B). For any ML model $M \in \text{Mod}_{\text{ML}}(\Sigma_{\text{LTL}})$ and an LTL formula $\varphi$, $T(M) \vDash_{\text{LTL}} \varphi \iff M \vDash [\varphi]_{\text{Trace}}$.

Validity checking, that is, checking whether a formula is true in all models, is enabled by a semantic equivalence theorem, which is a direct consequence of model equivalence theorems.

Theorem 11 (Semantic Equivalence). For any $\varphi \in \Phi_{\text{LTL}}$, $\vDash_{\text{LTL}} \varphi \iff \Gamma_{\text{LTL}} \models [\varphi]_{\text{Trace}}$.

The implication from left to right follows from Theorem 10, while the other implication from Theorem 4.

4 LTL Model Checking in Matching Logic

In this section we develop the theory of LTL model checking of systems specified in matching logic. The main idea is to come up with a matching logic theory whose models have two components, where one component represents the system (a Kripke structure as in Section 4.2) and the other component is used for LTL reasoning (as in Section 3). A Semantic Preservation by Model Gluing property (Lemma 19) ensures that the model consisting of the two parts (we call such model a glued model) behaves for the purposes of reasoning about one of its parts as if the other part were not present; specifically, a matching logic model created by gluing a Kripke-structure-like model with an LTL-like model satisfies all the Kripke structure and LTL axioms, and thus all the results for reasoning about the parts (e.g., those from Section 3) still apply. The two model parts are connected together using a linking theorem (Lemma 20), so that the LTL reasoning applies to the given Kripke structure. By connecting these three ideas we get a model checking theorem (Theorem 21), which states, intuitively, that an LTL formula holds in a Kripke structure specified by a matching logic theory iff its translation holds in a particular matching logic theory.

4.1 Kripke Structures

We use the standard definitions of Kripke structures.

Definition 12 (Kripke structure [20]). Given a set $AP$, a Kripke structure $K$ over $AP$ is a tuple $K = (S, S_0, \delta, L)$, where $S_0 \subseteq S$, $\delta : S \to 2^S$, and
$L : S \rightarrow 2^{AP}$. Let $KS(AP)$ (or just $KS$ if $AP$ is known) denote the class of all Kripke structures over $AP$.

**Definition 13.** The set $Traces(K) \subseteq P(AP)^\omega$ of traces of a Kripke structure $K = (S, S_0, \delta, L)$ contains a trace $\tau$ if and only if there exists a sequence $s_0 s_1 s_2 \ldots$ of states from $S$ such that $s_0 \in S_0$ and for all $i \geq 0$ it holds that $s_{i+1} \in \delta(s_i)$ and $\tau[i] = L(s_i)$.

**Definition 14.** We say that a Kripke structure satisfies an LTL formula iff all its traces satisfy the formula. Formally, we write $K \models_{\text{LTL}} \phi$ iff for all $\tau \in Traces(K)$, $\tau \models \phi$.

### 4.2 Kripke Structures in Matching Logic

In Spec. 4, we define a theory $\Gamma_{KS}$ whose models correspond to Kripke structure. Intuitively, we have there a set of states represented as inhabitants of the sort $\text{State}$ and a set of initial states represented as inhabitants of the sort $\text{InitState}$; the axiom ($KS_{\text{initSub}}$) says that the second is a subsort of the first. Next, there is the symbol $(\rightarrow)$ representing a transition relation; the axiom ($KS_{\text{NextS}}$) ensures that if the symbol is applied to a state, it returns a set of states. Additionally, we have a notation that allows us to write $a \rightarrow b$ whenever $b$ is among the successors of $a$. Finally, we assume one symbol $st_a$ for every atomic proposition $a \in AP$; these symbols are axiomatized by the axiom schema ($KS_{\text{AP}}$) to be matched by a sets of state; intuitively, these states are those in which the corresponding atomic proposition holds. We formalize the relationship between $\Gamma_{KS}$-models and Kripke structures by means of the pair of functions $KS_{\text{of}} : \text{Mod}_{\text{ML}}(\Gamma_{KS}) \rightarrow KS$ and $K_{\text{of}} : KS \rightarrow \text{Mod}_{\text{ML}}(\Gamma_{KS})$, defined in the appendix.

<table>
<thead>
<tr>
<th>Spec KS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Import:</strong></td>
</tr>
<tr>
<td><strong>Symbols:</strong></td>
</tr>
<tr>
<td><strong>Notations:</strong></td>
</tr>
<tr>
<td><strong>Axioms:</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Spec. 4: A theory of Kripke structures

### 4.3 Open Fragment of Matching Logic

We now present a syntactical fragment of matching logic that we dubbed an *open fragment*. What we mean by *open* is that with this fragment, models are *open to extension*, in the sense that adding more elements to a model does not change
the meaning of a pattern from this fragment. This is achieved by restricting the quantification and negation to their sorted version, so that, intuitively, a pattern cannot reach to the elements outside of the signature. More formally, we define the subsets of open data patterns and open predicate patterns, and prove what we call a Semantics Preservation by Model Extension theorem (Theorem 17), saying that (1) open data patterns match exactly the same elements in the extended model as in the original model, and (2) open predicate patterns hold in the extended model iff they hold in the original model. Conveniently, the open fragment is large enough to include all the axioms of $\Gamma_{LTL}$ and $\Gamma_{KS}$, and the translated LTL formulas. As a consequence, we can glue a model of $\Gamma_{LTL}$ and a model of $\Gamma_{KS}$ together, without breaking the matching logic formulas of interest.

Definition 15 (Open Fragment). Let $\Sigma$ be given such that $\{\text{def}, \text{inh}\} \subseteq \Sigma$. We define $\Psi_{\text{pred}}(\Sigma) \subseteq \Phi(\Sigma)$ to be the fragment of matching logic patterns generated from the nonterminal $\psi$ in the grammar below, and $\Phi_{\text{data}}(\Sigma) \subseteq \Phi(\Sigma)$ to be the fragment generated from the nonterminal $\phi$; $s$ ranges over symbols other than $\text{def}$ and $\text{inh}$.

$\psi ::= \bot \mid \text{def} \varphi \mid \varphi = \varphi \mid \varphi \subseteq \varphi \mid \exists x. x \in (\text{inh } s) \land \psi \mid \forall x. x \in (\text{inh } s) \rightarrow \psi$

$\varphi ::= \bot \mid x \mid X \mid s \mid \text{inh } s \mid (\text{inh } s) \land (\neg \varphi) \mid \varphi \land \varphi \mid \varphi \lor \psi \mid \exists x. x \in (\text{inh } s) \land \varphi \mid \mu X. \varphi$

where $\varphi$ is positive in $X$

The intuition behind the above definition is that open predicate pattern represent logical constraints, and are built by sorted first-order combinations of atomic constraints like definedness, equality, and inclusion; among data patterns (which represent data), an interesting connective is the filtering conjunction, which allows one to filter a data pattern through a predicate pattern: the result is always either the data pattern itself, or bottom. Intuitively, data patterns cannot reach to some model elements, because negation and quantification is guarded by the conjunction with the inhabitant set of some sort; open predicate patterns can be matched either by all elements, or by none. Therefore, open patterns cannot distinguish between models whose only difference is in “foreign” elements.

This intuition is captured by the construction we call model extension and the corresponding theorem. The following definition captures precisely what it means for us to extend a model with a set. The constructed model interprets definedness in the standard way, otherwise preserves the interpretation of the “old” model.

Definition 16 (Model Extension). Let $\Sigma \supseteq \{\text{def}, \text{inh}\}$. Let $(M, \{\sigma_M\}_{\sigma \in \Sigma}, \text{app}_M(\_\_, \_))$ be a $\Sigma$-model. Let $R$ be a set that is disjoint with $M$. Let $f_{R,M} : R \times M \rightarrow \mathcal{P}(M \cup R)$, $f_{M,R} : M \times R \rightarrow \mathcal{P}(M \cup R)$, $f_{R,R} : R \times R \rightarrow \mathcal{P}(M \cup R)$, $f_{\text{inh}} : R \rightarrow \mathcal{P}(M \cup R)$ be functions. We define a $\Sigma$-model $\text{mext}(M, R, f_{R,M}, f_{M,R}, f_{R,R}, f_{\text{inh}}) = (M', \{\sigma_{M'}\}_{\sigma \in \Sigma}, \text{app}_{M'})$ by

- $M' = M \cup R \cup \{\#\text{def}, \#\text{inh}\}$, where $\#\text{def}, \#\text{inh}$ are fresh;
- $\text{def}_{M'} = \{\#\text{def}\}$;
- $\text{inh}_{M'} = \{\#\text{inh}\}$;
• \( \sigma_{M'} = \sigma_M \) for any \( \sigma \in (\Sigma \setminus \{\text{def, inh}\}) \);
• \( \text{app}_{M'}(\#\text{def}, m) = M' \) for any \( m \in M' \);
• \( \text{app}_{M'}(\#\text{inh}, m) = \text{app}_{M}(\text{inh}_M, \{m\}) \) for any \( m \in M \);
• \( \text{app}_{M'}(\#\text{inh}, r) = f_{\text{inh}}(r) \) for any \( r \in R \);
• \( \text{app}_{M'}(r_1, r_2) = f_{R,R}(r_1, r_1) \) for any \( r_1, r_2 \in R \);
• \( \text{app}_{M'}(m, r) = f_{M,R}(m, r) \) for any \( r \in R \) and \( m \in M \);
• \( \text{app}_{M'}(r, m) = f_{R,M}(r, m) \) for any \( r \in R \) and \( m \in M \);
• \( \text{app}_{M'}(m_1, m_2) = \text{app}_{M}(m_1, m_2) \) for any \( m_1, m_2 \in M \);

**Definition 18** (Model Gluing). Let \( \Sigma_1, \Sigma_2 \) be two matching logic signatures such that \( \Sigma_1 \cap \Sigma_2 = \Sigma_c \), where \( \Sigma_c = \{\text{def, inh}\} \). Let \( M_1 \) be a \( \Sigma_1 \)-model and \( M_2 \) be a \( \Sigma_2 \) model such that the two have disjoint carrier sets. We define a glued model \( M_g = \text{glue}(M_1, M_2) \) by

1. defining the carrier set \( M_g = M_1 \cup M_2 \cup \{\#\text{def, #inh}\} \), where \#def, \#inh are fresh;
2. defining the interpretation of symbols by
   • \( \text{def}_{M_g} = \{\#\text{def}\} \);
   • \( \text{inh}_{M_g} = \{\#\text{inh}\} \);
   • \( s_{M_g} = s_{M_1} \) for any \( s \in \Sigma_1 \setminus \Sigma_c \);
   • \( s_{M_g} = s_{M_2} \) for any \( s \in \Sigma_2 \setminus \Sigma_c \);
3. and by defining the interpretation of application by
   • \( \text{app}_{M_g}(\#\text{def}, m) = M_g \) for any \( m \in M_g \);
   • \( \text{app}_{M_g}(\#\text{inh}, m_1) = \text{app}_{M_1}(\text{inh}_M, \{m_1\}) \) for any \( m_1 \in M_1 \);
   • \( \text{app}_{M_g}(\#\text{inh}, m_2) = \text{app}_{M_2}(\text{inh}_M, \{m_2\}) \) for any \( m_2 \in M_2 \);

**Theorem 17** (Semantics Preservation by Model Extension). Let \( M' \) be as constructed in Definition 176 from \( M \), and let \( M \) satisfies the definedness axiom. Then for any \( M \)-valuation \( \rho \) and any open data pattern \( \psi \in \Phi_{\text{data}}(\Sigma) \), it holds that \( |\psi|_{M', \rho} = |\psi|_{M, \rho} \); furthermore, for any open predicate pattern \( \psi \in \Psi_{\text{pred}}(\Sigma) \), it holds that

\[
|\psi|_{M', \rho} = \emptyset \iff |\psi|_{M, \rho} = \emptyset \quad \text{and} \quad |\psi|_{M', \rho} = M' \iff |\psi|_{M, \rho} = M .
\]

That is, open data patterns are interpreted the same in both models, and open predicate patterns hold in the new model iff they hold in the original model.

An immediate corollary is that the constructed model satisfies the same open theories as the original model (where by “open theory” we mean a theory containing only open predicate patterns).

A related notion to model extension is model gluing. Intuitively, we want to join two models into one such that the resulting model keeps some properties of the original models.

**Definition 18** (Model Gluing). Let \( \Sigma_1, \Sigma_2 \) be two matching logic signatures such that \( \Sigma_1 \cap \Sigma_2 = \Sigma_c \), where \( \Sigma_c = \{\text{def, inh}\} \). Let \( M_1 \) be a \( \Sigma_1 \)-model and \( M_2 \) be a \( \Sigma_2 \) model such that the two have disjoint carrier sets. We define a glued model \( M_g = \text{glue}(M_1, M_2) \) by

1. defining the carrier set \( M_g = M_1 \cup M_2 \cup \{\#\text{def, #inh}\} \), where \#def, \#inh are fresh;
2. defining the interpretation of symbols by
   • \( \text{def}_{M_g} = \{\#\text{def}\} \);
   • \( \text{inh}_{M_g} = \{\#\text{inh}\} \);
   • \( s_{M_g} = s_{M_1} \) for any \( s \in \Sigma_1 \setminus \Sigma_c \);
   • \( s_{M_g} = s_{M_2} \) for any \( s \in \Sigma_2 \setminus \Sigma_c \);
3. and by defining the interpretation of application by
   • \( \text{app}_{M_g}(\#\text{def}, m) = M_g \) for any \( m \in M_g \);
   • \( \text{app}_{M_g}(\#\text{inh}, m_1) = \text{app}_{M_1}(\text{inh}_M, \{m_1\}) \) for any \( m_1 \in M_1 \);
   • \( \text{app}_{M_g}(\#\text{inh}, m_2) = \text{app}_{M_2}(\text{inh}_M, \{m_2\}) \) for any \( m_2 \in M_2 \);
• $\text{app}_{M_g}(m_1, m_2) = \text{app}_{M_1}(m_1, m_2)$ for any $m_1, m_2 \in M_1$;
• $\text{app}_{M_g}(m_1, m_2) = \text{app}_{M_2}(m_1, m_2)$ for any $m_1, m_2 \in M_2$;
• and $\text{app}_{M_g}(\cdot, \cdot) = \emptyset$ otherwise.

It is easy to see that gluing is a special instance of model extension followed by an extension of the signature: one can take $R = M_2$, $f_{\text{inh}}(m_2) = \text{appext}_{M_2}(\text{inh}_{M_2}, \{ m_2 \})$, $f_{R,R} = \text{app}_{M_2}$, $f_{M,R}(m,r) = \emptyset$, and $f_{R,M}(r,m) = \emptyset$.

Therefore, gluing preserves the semantics of data and predicate patterns.

**Lemma 19 (Semantic Preservation by Model Gluing).** For any $i \in \{1, 2\}$, let $M_i$ be a $\Sigma_i$-model satisfying the definedness axiom, let $\rho_i$ be an $M_i$-valuation, and let $M_g = \text{glue}(M_1, M_2)$. Then for any $\varphi \in \Phi_{\text{data}}(\Sigma_i)$, $|\varphi|_{M_g, \rho_1} = |\varphi|_{M_i, \rho_i}$, and for any $\psi \in \Psi_{\text{pred}}(\Sigma_i)$, $|\psi|_{M_g, \rho_1} = \emptyset \iff |\psi|_{M_i, \rho_i} = \emptyset$ and $|\psi|_{M_g, \rho_1} = M_g \iff |\psi|_{M_i, \rho_i} = M_i$.

### 4.4 Linking Kripke Structures and LTL

It is easy to see that axioms for LTL and Kripke structures fall into the open fragment of matching logic. Therefore, what we have so far allows us to glue a $\Gamma_{\text{KS}}$-model together with a $\Gamma_{\text{LTL}}$-model, extend the glued model with other elements, and reason about the resulting model similarly to the original models. However, to perform LTL model checking of a particular Kripke structure, one has to connect the two parts together; that is, to filter out models in which the represented LTL trace does not correspond to any path in the represented Kripke structure. We connect the two in this section.

**Spec LINK** Import: LINKPRELIM

Notations:

- $St \equiv \text{State}$
- $Ts \equiv \text{TraceSuffix}$
- $(st, tr) \equiv (\text{psts } st \ tr)$
- $st \sim_{\text{AP}} tr \equiv \bigwedge_{a \in \text{AP}} (st \subseteq st_a \leftrightarrow tr \subseteq tr_a)$
- $(st, tr) \sim_{3} X \equiv \exists st' : St \sim st' \land (st', \bar{tr}) \in X$
- $\text{Corr} \equiv \nu X. \exists st : St. \exists tr : Ts. (st, tr) \land st \sim_{\text{AP}} tr \land (st, tr) \sim_{3} X$

Axioms:

- (LTLGLUE) $\exists s_0 : \text{InitState}. (s_0, \text{Trace}) \in \text{Corr}$

Spec. 5: A theory linking Kripke structures and LTL reasoning

The linking theory $\Gamma_{\text{LINK}}$ has two parts: $\Gamma_{\text{LINKPRELIM}}$ only axiomatizes pairs (in a standard way, e.g. [6]) which are then used by the rest of $\Gamma_{\text{LINK}}$; we therefore show only the second part here (in Spec. 5), and an interested reader will find the
first part in the appendix. The purpose of the notations in the specification is to define a correspondence relation $\text{Corr}$ between states of the Kripke structure and trace suffixes used for LTL reasoning. The intuitive meaning of $\text{Corr}$ is that a state $st$ is in relation with a trace suffix $tr$ iff the two agree on atomic propositions and some successor of $st$ is in relation with the tail of $tr$. The axiom (LTLGLUE) then ensures that the full trace represented by the LTL component of the model is in relation with some initial state of the Kripke structure. The correctness of this construction is then stated as the following lemma.

**Lemma 20** (Correctness of $\Gamma_{\text{LINK}}$). For any $(\Gamma^{\text{LTL}} \cup \Gamma^{\text{KS}} \cup \Gamma^{\text{LINKPRELIM}})$-model $M$, $M \models \Gamma^{\text{LINK}} \iff T(M) \in \text{Traces(KSof}(M))$.

The right-to-left implication is the harder one; this direction is also the reason why we have only one axiom in $\Gamma^{\text{LINK}} \setminus \Gamma^{\text{LINKPRELIM}}$, it would not be easy to ensure other axioms in a model about which we know so little. Note that we have to assume the axioms about pairs in $\Gamma^{\text{LINKPRELIM}}$; we could not prove them just from the right side of the equivalence.

### 4.5 Model Checking

Now we are equipped to state and prove the main result of the paper: the model checking theorem. Let $\Gamma^{\text{ltlmc}} = \Gamma^{\text{LTL}} \cup \Gamma^{\text{LINK}}$. We assume a Kripke structure $S$ and a theory $\Gamma^{S} \supseteq \Gamma^{\text{KS}}$, whose axioms are open predicates (in the above extension/gluing sense), satisfied by exactly those matching logic models that projects to Kripke structures isomorphic to $S$. Then:

**Theorem 21.** $S \models_{\text{LTL}} \varphi \iff \Gamma^{S} \cup \Gamma^{\text{MC}} \models [\varphi]_{\text{Trace}}$.

### 5 Related Work

We already mentioned in the introduction that in [7], matching logic captures LTL tautologies. Another related work is in [11], where the authors discuss model checking of finite systems specified in rewriting logic using the Maude tool; that work also shows how to perform model checking of programs whose language is formalized in the rewriting logic. Admittedly, their work is more developed than ours: for example, their Maude implementation has an implementation of an automata-based LTL model checking algorithm supporting predicate abstraction. More information about the rewriting-logic approach to language semantics can be found, e.g., in [19].

More generally, the book “Handbook of Model Checking” [10] contains a chapter on combining model checking and deduction [21]. Another source on LTL model checking is the survey [18].

### References


A Appendix

This section contains proofs of the theorems from the main text. Some generic matching logic theorems have also been formalized in Coq, as part of the Matching Logic in Coq project [2]. These proofs can be found in [1], and we always refer to them. In some cases, usually when the proof is technical, long, or uninteresting, we feel free to omit the proof from the appendix and provide only a reference to the Coq formalization.

A.1 Proofs About Matching Logic

A.1.1 Monotonicity of pattern interpretation

The main theorem of this section, Lemma 25, has been formalized in [1], file monotonic.v.

Lemma 22. Let $M$ be a set and $F_1, F_2 : \mathcal{P}(M) \rightarrow \mathcal{P}(M)$ monotonic function such that for any $A \subseteq M$, $F_1(A) \subseteq F_2(A)$. Then $\mu F_1 \subseteq \mu F_2$.

Proof. Since $\mu F_1$ is the least prefixpoint of $F_1$, it is enough to show that $\mu F_2$ is a prefixpoint of $F_1$; i.e., that $F_1(\mu F_2) \subseteq \mu F_2$. But that holds by $F_1(\mu F_2) \subseteq F_2(\mu F_2) = \mu F_2$. $\square$

Definition 23. A blacklist is a pair $(B_P, B_N)$ of sets of set variables $B_P, B_N \subseteq SV$. We say that a ML pattern $\varphi$ respects blacklist $(B_P, B_N)$ iff no variable $V_P \in B_P$ occurs positively in $\varphi$ and no variable $V_N \in B_N$ occurs negatively in $\varphi$.

Lemma 24. If $\varphi$ respects blacklist $(B_P, B_N)$, then for any $\Sigma$-model $(M, app_M, \{\sigma_M\}_{\sigma \in \Sigma})$ and any $M$-valuation $\rho$, the function $F_{\varphi, \rho}^M : \mathcal{P}(M) \rightarrow \mathcal{P}(M)$ defined by $F_{\varphi, \rho}^M(A) = |\varphi|_{M, \rho}[A/V]$ for any $A \subseteq M$ is antimonotonic for any $V \in B_P$ and monotonic for any $V \in B_N$.

Proof. Let $A_1 \subseteq A_2 \subseteq M$ We will proceed by structural induction on $\varphi$.

- $\varphi \equiv x$:
  
  $F_{\varphi, \rho}^M(A_1) = |x|_{M, \rho}[A_1/V]$
  
  $= \{\rho(x)\}$
  
  $= |x|_{M, \rho}[A_2/V]$
  
  $= F_{\varphi, \rho}^M(A_2)$

- $\varphi \equiv X$:
  
  Let $V \in B_P$. Since $X$ occurs positively in $X$ and $X$ respects $(B_N, B_P)$, it follows that $X \neq V$. Then
  
  $F_{\varphi, \rho}^M(A_1) = |X|_{M, \rho}[A_1/V]$
  
  $= \rho(X)$
  
  $= |X|_{M, \rho}[A_2/V]$
  
  $= F_{\varphi, \rho}^M(A_2)$.
Let $V \in B_N$. Then either $X \neq V$, and $F_{\varphi,V}^M(A) = \rho(X) = F_{\varphi,V}^M(A_2)$ as in the previous case, or $X = V$. Then

$$F_{\varphi,V}^M(A_1) = |X|_{M,\rho}(A_1/V) = |X|_{M,\rho}(A_1/X) = A_1 \subseteq A_2 = |X|_{M,\rho}(A_2/X) = |X|_{M,\rho}(A_2/V) = F_{\varphi,V}^M(A_2).$$

- $\varphi \equiv \bot$ - trivial.
- $\varphi \equiv \sigma$ - trivial.
- $\varphi \equiv \varphi_1 \rightarrow \varphi_2$ - then $\varphi_2$ respects $(B_P, B_N)$ and $\varphi_1$ respects $(B_N, B_P)$.

- Let $V \in B_P$. By the induction hypothesis, $F_{\varphi_2,V}^M$ is antimonotonic and $F_{\varphi_1,V}^M$ is monotonic. Therefore,

$$|\varphi_2|_{M,\rho}(A_1/V) = F_{\varphi_2,V}^M(A_1) \supseteq F_{\varphi_2,V}^M(A_2) = |\varphi_2|_{M,\rho}(A_2/V)$$

and

$$|\varphi_1|_{M,\rho}(A_1/V) = F_{\varphi_1,V}^M(A_1) \subseteq F_{\varphi_1,V}^M(A_2) = |\varphi_1|_{M,\rho}(A_2/V).$$

Therefore,

$$|\varphi_1|_{M,\rho}(A_1/V) \setminus |\varphi_2|_{M,\rho}(A_1/V) \subseteq |\varphi_1|_{M,\rho}(A_2/V) \setminus |\varphi_2|_{M,\rho}(A_2/V)$$

and

$$F_{\varphi,V}^M(A_1) = |\varphi_1 \rightarrow \varphi_2|_{M,\rho}(A_1/V) = M \setminus (|\varphi_1|_{M,\rho}(A_1/V) \setminus |\varphi_2|_{M,\rho}(A_1/V)) \supseteq M \setminus (|\varphi_1|_{M,\rho}(A_2/V) \setminus |\varphi_2|_{M,\rho}(A_2/V)) = |\varphi_1 \rightarrow \varphi_2|_{M,\rho}(A_2/V) = F_{\varphi,V}^M(A_2).$$

- Let $V \in B_N$. This case is proved similarly.

- $\varphi \equiv \varphi_1 \varphi_2$ - follows from $app_M$ being a pointwise extension.
\[ \phi \equiv \exists x. \phi' \]

- Let \( V \in B_P \). Then

\[
\mathcal{F}_{\phi',V}(A_1) = |\exists x. \phi'|_{M,\rho[A_1/V]} \\
= \bigcup_{m \in M} |\phi'|_{M,\rho[m/x][A_1/V]} \\
\geq \bigcup_{m \in M} |\phi'|_{M,\rho[m/x][A_2/V]} \\
= \bigcup_{m \in M} |\phi'|_{M,\rho[m/x][A_2/V]} \\
= |\exists x. \phi'|_{M,\rho[A_1/V]} \\
= \mathcal{F}_{\phi',V}(A_2)
\]

where the middle inequality holds because by the induction hypothesis,
\( \mathcal{F}_{\phi',V}[m/x] \) is antimonotonic.

- Let \( V \in B_N \). Proved similarly to the previous case.

\[ \phi \equiv \mu X. \phi' \] - Since \( \phi' \) does not contain a negative occurrence of \( X \), \( \phi' \) respects blacklist \((B_P, B_N \cup \{X\})\).

- Let \( V \in B_P \). We need to prove the inequality in

\[
\mathcal{F}_{\phi',X}(A_1) = |\mu X. \phi'|_{M,\rho[A_1/V]} \\
= \mu \mathcal{F}_{\phi',X}[A_1/V] \\
\geq \mu \mathcal{F}_{\phi',X}[A_2/V] \\
= |\mu X. \phi'|_{M,\rho[A_2/V]} \\
= \mathcal{F}_{\phi',X}(A_2)
\]

(The fixpoint exists because by the induction hypothesis, the functions are monotone in \( X \).) By Lemma 22, it is enough to show that for any \( B \in M \), \( \mathcal{F}_{\phi',X}[A_1/V](B) \supseteq \mathcal{F}_{\phi',X}[A_2/V](B) \). If \( X \neq V \), then using the induction hypothesis,

\[
\mathcal{F}_{\phi',X}[A_1/V](B) = |\phi'|_{M,\rho[A_1/V][B/X]} \\
= |\phi'|_{M,\rho[B/X][A_1/V]} \\
\geq |\phi'|_{M,\rho[B/X][A_2/V]} \\
= |\phi'|_{M,\rho[A_2/V][B/X]} \\
= \mathcal{F}_{\phi',X}[A_1/V](B)
\]

On the other hand, if \( X = V \), then

\[
\mathcal{F}_{\phi',X}[A_1/V](B) = |\phi'|_{M,\rho[B/V]} = \mathcal{F}_{\phi',X}(B).
\]
Proof. Since $X$ has no negative occurrence in $\varphi$, it follows that $\varphi$ respect blacklist $(\emptyset, \{X\})$ and therefore by Lemma 24, $F_{\varphi, X}^{\rho, \rho}$ is monotonic. □

A.2 Proofs About the LTL Embedding in Matching Logic

We define a formula translation function $L2M' : \Phi_{\text{TL}} \to \text{Pattern}(\Sigma_{\text{TL}})$ from LTL formulas to matching logic formulas. The translation function is straightforward, directly mapping LTL constructs to symbols and notations of the specification LTL:

- $L2M'(a) = a$ for all atomic propositions $a$;
- $L2M'(!\varphi) = !_{\text{TL}}L2M'(\varphi)$;
- $L2M'(\varphi \land \psi) = L2M'(\varphi) \land L2M'(\psi)$;
- $L2M'(\varphi \lor \psi) = \cup L2M'(\varphi)$; and
- $L2M'(\psi_1 U \psi_2) = L2M'(\psi_1) U L2M'(\psi_2)$.

Lemma 26. For any model $M$ satisfying (PREV) and any $m_1, m_2 \in M$,

$m_2 \in \text{apext}_M(\hat{\varphi}_M, \{m_1\}) \iff m_1 \in \text{apext}_M(\varphi_M, \{m_2\}) \land m_2 \in M_{\text{TraceSuffix}}$.

Proof of Lemma 26 Let $\rho(x) = m_1$. Then $m_2 \in \text{apext}_M(\hat{\varphi}_M, \{m_1\}) = [\hat{\varphi}]_\rho = [\exists y. y \lor (x \in cy)]_\rho$ (by the axiom (PREV)) = $\{m' \in M_{\text{TraceSuffix}} \mid M = [x \in cy]_\rho [m'/y] \} \land m_2 \in M_{\text{TraceSuffix}}$ if $m_1 \in \text{apext}_M(\varphi_M, \{m_2\})$.

Definition 27. Let $(M, \text{app}(\cdot, \cdot), \{\sigma_M\}_{\sigma \in \Sigma_{\text{TL}}})$ be a $\Gamma_{\text{TL}}$-model. Let us define:

- a partial function $M_5 : M_{\text{TraceSuffix}} \rightarrow M_{\text{TraceSuffix}}$ defined by $M_5(m) = m'$ whenever $m'$ is the unique element satisfying $\{m'\} = \text{apext}_M(\hat{\varphi}_M, \{m\})$; and
- a (total) function $M_6 : M_{\text{TraceSuffix}} \rightarrow M_{\text{TraceSuffix}}$ defined by $M_6(m) = m'$, where $m'$ is the unique element satisfying $\{m'\} = \text{apext}_M(\hat{\varphi}_M, \{m\})$ for all $m \in M$.

Lemma 28. Definition 27 is well-formed, and $M_5$ and $M_6$ are injective.

Proof of Lemma 28 The definition is well-formed by axioms (PREVFUN) and (NEXTPFUN); see interp_total_function and interp_partial_function from file Sorts_Semantics.v of [1] for more detail. The injectivity follows from the fact that the two are inverses (Lemma 26). □
Proof of Lemma 2: By structural induction on $\varphi$:

- $\varphi \equiv a$ where $a \in \text{AP} - |\text{tr}_a|_M \subseteq M_{\text{TraceSuffix}}$ by the axiom (\text{ATOMICPROP}).

- $\varphi \equiv \lnot \varphi'$ - $|L_2M'(\lnot \varphi)|_M = |\text{TraceSuffix}| \land \lnot L_2M'(\varphi)|_M = M_{\text{TraceSuffix}} \cap |\lnot L_2M'(\varphi)|_M \subseteq |\text{TraceSuffix}|$.

- $\varphi \equiv \varphi_1 \land \varphi_2$ - $|L_2M'(\varphi_1 \land \varphi_2)|_M = |L_2M'(\varphi_1)|_M \land |L_2M'(\varphi_2)|_M \subseteq M_{\text{TraceSuffix}} \cap M_{\text{TraceSuffix}} = M_{\text{TraceSuffix}}$.

- $\varphi \equiv \varphi_1 \lor \varphi_2$ - $|L_2M'(\varphi_1 \lor \varphi_2)|_M = |\text{TraceSuffix}| \lor |L_2M'(\varphi_1)|_M \subseteq M_{\text{TraceSuffix}} \cap M_{\text{TraceSuffix}} = M_{\text{TraceSuffix}}$.

- $\varphi \equiv \varphi_1 \mathrel{U} \varphi_2$ - We need to show that $|L_2M'(\varphi_1 \mathrel{U} \varphi_2)|_M = |\text{TraceSuffix}| \lor |L_2M'(\varphi_1)|_M \subseteq M_{\text{TraceSuffix}} \cap M_{\text{TraceSuffix}} = M_{\text{TraceSuffix}}$ holds using the induction hypothesis.

Definition 29. We define a model translation function $M(\cdot) : \text{Mod}_{\text{TL}} \to \text{Mod}_{\text{LTL}}(\text{LTL})$ as follows. For an LTL model, i.e., a trace $\tau \in \text{Mod}_{\text{LTL}}$, let the carrier set $M(\tau)$ be the disjoint union of $\mathbb{N}_{\geq 1}$ (the set of positive natural numbers) and

$$\{\#\text{def}, \#\text{inh}, \#\text{Trace}, \#\text{TrSuf}, \#\text{next}, \#\text{prev}\}$$

(a set of six distinguished elements). We let $\text{def}_{M(\tau)} = \{\#\text{def}\}$, $\text{inh}_{M(\tau)} = \{\#\text{inh}\}$, $\text{Trace}_{M(\tau)} = \{\#\text{Trace}\}$, $\text{TrSuf}_{M(\tau)} = \{\#\text{TrSuf}\}$, $\sigma_{M(\tau)} = \{\#\text{next}\}$, $\delta_{M(\tau)} = \{\#\text{prev}\}$, and $\text{tr}_{a,M(\tau)} = \{n \in \mathbb{N}_{\geq 1} \mid a \in \tau[n]\}$ for any $a \in \text{AP}$, and define the application as follows:

1. $\text{app}_{M(\tau)}(\#\text{def}, m) = M(\tau)$ for any $m \in M(\tau)$;

2. $\text{app}_{M(\tau)}(\#\text{inh}, \#\text{Trace}) = \{1\}$;

3. $\text{app}_{M(\tau)}(\#\text{inh}, \#\text{TrSuf}) = \mathbb{N}_{\geq 1}$;

4. $\text{app}_{M(\tau)}(\#\text{next}, 1) = \emptyset$;

5. $\text{app}_{M(\tau)}(\#\text{next}, (n + 1)) = \{n\}$ for all $n \in \mathbb{N}_{\geq 1}$;

6. $\text{app}_{M(\tau)}(\#\text{prev}, n) = \{n + 1\}$ for all $n \in \mathbb{N}_{\geq 1}$;
Proposition 30. Function $M(\_)$ is well-defined, i.e., $M(\tau) \models \Gamma_{LTL}$ for any LTL model $\tau \in Mod_{LTL}$.

Proof of Proposition 30. We will prove the axioms one by one. Let $\rho: (EV \cup SV) \rightarrow (M(\tau) \cup \mathcal{P}(M(\tau)))$ be an $M(\tau)$-valuation. Then:

- (Definedness) -
  \[
  [x]_\rho = app_{M(\tau)}(\#def, [x]_\rho) = app_{M(\tau)}(\#def, \rho(x)) = app_{M(\tau)}(\#def, \rho(x)) = M(\tau)
  \]

- (Prev) - $M(\tau) = [\bar{o}x = \exists y : TraceSuffix. y \land (x \in oy)]_\rho$ iff $[\bar{o}x]_\rho = [\exists y : TraceSuffix. y \land (x \in oy)]_\rho$, which holds because
  \[
  [\bar{o}x]_\rho = app_{M(\tau)}(\#prev, \rho(x)) = \{\rho(x) + 1\} = \{\rho(x) + 1 | \rho(x) \in app_{M(\tau)}(\#next, \rho(x) + 1)\} = \{m + 1 \mid \rho(x) \in app_{M(\tau)}(\#next, m + 1)\} = \{m \mid \rho(x) \in app_{M(\tau)}(\#next, m)\} = [\exists y : TraceSuffix. y \land (x \in oy)]_\rho.
  \]

- (Trace) - $[\exists x. \llbracket Trace \rrbracket = x]_\rho \subseteq [\llbracket Trace \rrbracket = x]_{\rho[1/x]} = M(\tau)$, because $[\llbracket \rho[1/x](\llbracket Trace \rrbracket) = \{1\}$ and $[\llbracket \rho[1/x](x) = \{1\}$.

- (TraceSuffix) - We want to prove that $[\llbracket TraceSuffix \rrbracket = \mu X. [\llbracket Trace \rrbracket \lor \bar{o}X]_\rho = M(\tau)$. Since $[\llbracket TraceSuffix \rrbracket]_\rho = N_{\geq 1}$, we need to show that $N_{\geq 1} = \mu F$, where $F(A) = [\llbracket Trace \rrbracket]_\rho \cup [\bar{o}X]_{\rho[A/X]}$; i.e., $N_{\geq 1}$ is the least fixpoint of $F$. First, it is a fixpoint:
  \[
  F(N_{\geq 1}) = \{1\} \cup [\exists y. X \in oy]_{\rho[N_{\geq 1}/X]} = \{1\} \cup \{m \in M(\tau) \mid [X \in oy]_{\rho[N_{\geq 1}/X]}[m/y] = M(\tau)\} = \{1\} \cup \{m \in M(\tau) \mid \exists m' \in N_{\geq 1}. m' \in app_{M(\tau)}(\#next, m)\} = \{1\} \cup \{m + 1 \mid m \in N_{\geq 1}\} = N_{\geq 1}.
  \]
  It is the least fixpoint. Let $A = F(A)$. We want to show that $N_{\geq 1} \subseteq A$. We will proceed by induction.

- $\{1\} = app_{M(\tau)}(\#inh, \#Trace) = [\llbracket Trace \rrbracket]_\rho \subseteq F(A) = A$.
  - Assuming $n \in N_{\geq 1}$, we will show that $n + 1 \in A = F(A)$. It holds if $n + 1 \in [\bar{o}X]_{\rho[A/X]}$, iff $n + 1 \in \bar{o}M(\tau)(A)$ iff $\exists m \in A, n + 1 \in \bar{o}M(\tau)(m)$ iff $\exists m \in A, m \in M(\tau), (n + 1)$, which holds by the construction and choice $m = n$. 

24
• \((\text{INIT})\) - straightforward.

• \((\text{PREVFUN})\) and \((\text{NEXTPFUN})\) - since application on \(#\text{prev}\) and \(#\text{next}\) behaves as a function (resp. partial function); see \texttt{interp\_total\_function}\ and \texttt{interp\_partial\_function}\ from file \texttt{Sorts\_Semantics.v} of \ref{section:background} for more detail.

• \((\text{ATOMICPROP})\) - \(|\text{tr}_a| \subseteq \llbracket \text{TraceSuffix} \rrbracket |_{\rho} = \mathcal{M}(\tau)\) by construction.

\[\Box\]

\textit{Proof of Lemma 3.}\ We will proceed by structural induction on \(\varphi\).

• \(\varphi \equiv a - \tau, i \models_{\text{TL}} a\) iff \(\models_{\text{TL}} a\) iff \(i \in \text{tr}_{a,\mathcal{M}(\tau)}\) (by the construction of \(\mathcal{M}(\tau)\)) iff \(i \in |\text{tr}_a|_{\mathcal{M}(\tau)}\) (by the definition of \(\models\)).

• \(\varphi \equiv \neg \varphi' - \tau, i \models_{\text{TL}} \neg \varphi'\) iff \(\neg \varphi, i \not\models_{\text{TL}} \varphi'\) (by the definition of \(\models_{\text{TL}}\)) iff \(i \not\in |\mathcal{L} \mathcal{M}(\varphi'),|_{\mathcal{M}(\tau)}\) (by the induction hypothesis) iff \(i \not\in |\neg \mathcal{L} \mathcal{M}(\varphi')|_{\mathcal{M}(\tau)} \cap \mathcal{M}(\tau) \text{TraceSuffix}\) (because \(i \in \mathcal{M}(\tau) \text{TraceSuffix}\) by the construction of \(\mathcal{M}(\tau)\)) iff \(i \in |\mathcal{L} \mathcal{M}(\neg \varphi')|_{\mathcal{M}(\tau)}\) (by desugaring notations).

• \(\varphi \equiv \varphi_1 \land \varphi_2 - \tau, i \models_{\text{TL}} \varphi_1 \land \varphi_2\) iff \(\varphi_1, \varphi_2 \models_{\text{TL}} \varphi_2\) (by the definition of \(\models_{\text{TL}}\)) iff \(i \in |\mathcal{L} \mathcal{M}(\varphi_1)|_{\mathcal{M}(\tau)} \cap \mathcal{M}(\tau) \text{TraceSuffix}\) (by the induction hypothesis) iff \(i \in |\mathcal{L} \mathcal{M}(\varphi_2)|_{\mathcal{M}(\tau)} \cup |\mathcal{L} \mathcal{M}(\varphi_2)|_{\mathcal{M}(\tau)}\) iff \(i \in \mathcal{L} \mathcal{M}(\varphi_1) \land \mathcal{L} \mathcal{M}(\varphi_2)|_{\mathcal{M}(\tau)}\) iff \(i \in \mathcal{L} \mathcal{M}(\varphi_1 \land \varphi_2)|_{\mathcal{M}(\tau)}\).

• \(\varphi \equiv \varphi \lor \varphi' - \tau, i \models_{\text{TL}} \varphi \lor \varphi'\) iff \(\varphi, \varphi' \models_{\text{TL}} \varphi'\) (by the definition of \(\models_{\text{TL}}\)) iff \(i \in \cup \{app_{\mathcal{M}(\tau)}(\#\text{next}, \varphi_j) \mid j \in \mathcal{L} \mathcal{M}(\varphi_j)|_{\mathcal{M}(\tau)}\}\) iff \(i \in \mathcal{L} \mathcal{M}(\varphi) \lor \mathcal{L} \mathcal{M}(\varphi')|_{\mathcal{M}(\tau)}\).

• \(\varphi \equiv \varphi_1 U \varphi_2 - \text{Since \(\varphi_1 \land \varphi_2\) is a notation for \(X \land \rho\land \varphi_2\) in ML, we have that \(\mathcal{L} \mathcal{M}(\varphi_1 U \varphi_2)|_{\mathcal{M}(\tau)} = \mu F_U\), where \(F_U : \mathcal{P}(\mathcal{M}(\tau)) \to \mathcal{P}(\mathcal{M}(\tau))\) is defined by \(F_U(A) = |\mathcal{L} \mathcal{M}(\varphi_1)|_{\mathcal{M}(\tau)} \cup \{\mathcal{L} \mathcal{M}(\varphi_1)|_{\mathcal{M}(\tau)} \cap (\text{appext}_{\mathcal{M}(\tau)}(\#\text{next}, A))\}\). Because \(\mu F_U = F_U(\mu F_U)\), i.e. \(\mu F_U\) is a fix-point of \(F_U\), we can expand \(\mathcal{L} \mathcal{M}(\varphi_1 U \varphi_2)|_{\mathcal{M}(\tau)}\) to \(\mathcal{L} \mathcal{M}(\varphi_1)|_{\mathcal{M}(\tau)} \cup (\text{appext}_{\mathcal{M}(\tau)}(\#\text{next}, \mathcal{L} \mathcal{M}(\varphi_1)|_{\mathcal{M}(\tau)})\).

For \(\tau, i \models_{\text{TL}} \varphi_1 U \varphi_2\) implies \(i \in |\mathcal{L} \mathcal{M}(\varphi_1 U \varphi_2)|_{\mathcal{M}(\tau)}\), we first prove a stronger statement:

\textbf{Claim 31.} \textit{For all} \(m, n \in \mathbb{N}\) \textit{such that} \(n \leq m\),

\((\tau, m \models_{\text{TL}} \varphi_2 \land \forall o < n. \tau, m - o - 1 \models_{\text{TL}} \varphi_1)\)

\(\implies \forall p \leq n. (m - p) \in |\mathcal{L} \mathcal{M}(\varphi_1 U \varphi_2)|_{\mathcal{M}(\tau)}\).
Proof. By induction on $n$.

- $n = 0$ - Assuming $\tau, m \vDash_{\text{LTL}} \phi_2$, by the (outer) induction hypothesis $m \in \lfloor L2M'(\phi_2)|_{M(\tau)} \rfloor$, and from [1] it follows that $(m - 0) \in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$.

- $n > 0$ - Assuming the induction hypothesis

$$(\tau, m \vDash_{\text{LTL}} \phi_2 \land \forall o < n - 1, \tau, m - o - 1 \vDash_{\text{LTL}} \phi_1)$$

$$\Rightarrow \forall p \leq n - 1, (m - p) \in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$$

and assuming $\tau, m \vDash_{\text{LTL}} \phi_2$ and $\forall o < n, \tau, m - o - 1 \vDash_{\text{LTL}} \phi_2$ it follows that $\forall p \leq n - 1, (m - p) \in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$. It remains to prove the case when $p = n$, i.e. $(m - n) \in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$. Because of the equation [1] it is enough to prove that $(m - n) \in \lfloor L2M'(\phi_1)|_{M(\tau)} \rfloor$ and $(m - n) \in \text{appext}_{M(\tau)}(\{\#\text{next}\}, L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor)$. For the former, we use the outer induction hypothesis and the assumption $\tau, m - (n - 1) - 1 \vDash_{\text{LTL}} \phi_1$, while the latter holds because by the definition of $\text{appext}_{M(\tau)}(\{\#\text{next}\}, L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor)$, we first prove a stronger statement:

Claim 32. For all $m, n \in \mathbb{N}$ such that $n \leq m$,

$$(\tau, m \not\vDash_{\text{LTL}} \phi_1 U \phi_2) \land \forall o \leq n, \tau, m - o \not\vDash_{\text{LTL}} \phi_2)$$

$$\Rightarrow \forall p \leq n, (m - p) \not\in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor.$$  

Proof. By induction on $n$.

- $n = 0$ - Assuming $\tau, m \not\vDash_{\text{LTL}} \phi_1$ and $\tau, m - 0 \not\vDash_{\text{LTL}} \phi_1$, by the (outer) induction hypothesis it holds that $m \not\in \lfloor L2M'(\phi_1)|_{M(\tau)} \rfloor$ and $m \not\in \lfloor \phi_2|_{M(\tau)} \rfloor$. But then by [1], $m \not\in \lfloor L2M'(\psi_1 U \psi_2)|_{M(\tau)} \rfloor$.

- $n > 0$ - Assume $\tau, m \not\vDash_{\text{LTL}} \phi_1$ and $\forall o \leq n, \tau, m - o \not\vDash_{\text{LTL}} \phi_2$. From the inner induction hypothesis it follows that $\forall p \leq n - 1, (m - p) \not\in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$. It remains to prove the case where $p = n$, that $(m - n) \not\in \lfloor L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor$. Since $\tau, m - n \not\vDash_{\text{LTL}} \psi_2$, from the outer induction hypothesis it follows that $(m - n) \not\in \lfloor L2M'(\phi_2)|_{M(\tau)} \rfloor$. By [1], it is now enough to prove that $(m - n) \not\in \text{appext}_{M(\tau)}(\{\#\text{next}\}, L2M'(\phi_1 U \phi_2)|_{M(\tau)} \rfloor)$, which is true if $(m - (n - 1)) \not\in \lfloor L2M'(\psi_1 U \psi_2)|_{M(\tau)} \rfloor$, which holds by the (inner) induction hypothesis.
Now assume \( \tau, i \not\models_{LTL} \varphi_1 \cup \varphi_2 \). By definition of \( \models_{LTL} \), either there is no \( j \geq i \) satisfying \( \tau, j \models_{LTL} \psi_2 \), in which case 0 is a fixpoint of \( F_U \), or strictly before first such \( j \) there exists some \( k, i \leq k < j \) satisfying \( \tau, k \not\models_{LTL} \varphi_1 \) (and by choice of \( j \) also \( \tau, k \not\models_{LTL} \varphi_2 \)). Then we can use \( k \) and \( k - i \) as parameters \( m \) and \( n \) of the claim above and by the choice \( p = k - i \) get \( i = k - (k - i) \not\in |L_2M'(\varphi_1 \cup \varphi_2)|_{M(\tau)}. \)

**Proof of Lemma 33.** Let \( M \) be a model of \( \Gamma^{LTL} \), and \( m \in M_{TraceSuffix} \). A sequence \( m_1, m_2, \ldots, m_n = m \) where \( m_i \in M \) such that \( m_1 = M_{Trace} \) and \( m_{i+1} = M_0(m_i) \) for all \( 1 \leq i < n \) is called an initial sequence of \( m \). For every \( m \) there exists exactly one such sequence, and we define a function \( dist_M^{LTL} : M_{TraceSuffix} \to \mathbb{N}_{\geq 1} \) by \( dist_M^{LTL}(m) = n \).

**Proof of Lemma 33.** A more general version of this lemma has been proved as patt_ind_gen_simpl and witnessed_elements_unique_seq in file FixpointReasoning.v of [1].

For existence, the axiom \( (TraceSuffix) \) ensures that \( M_{TraceSuffix} = \mu F \), where \( F(A) = M_{Trace} \cup M_0(A) \). Define:

\[
\xi = \{ m \mid \exists n \in \mathbb{N}_{\geq 1}, \exists m_1, \ldots, m_n, m_1 = M_{Trace} \\
\land m_n = m \land \forall 1 \leq i < n, m_{i+1} = M_0(m_i) \}
\]

and since \( F(\xi) \subseteq \xi \), i.e. \( \xi \) is a prefix point of \( F \), from the Knaster-Tarski theorem it follows that \( M \subseteq \xi \), i.e. for every \( m \in M \) there exists an appropriate sequence.

For uniqueness, let there be two such sequences, \( m_1, \ldots, m_n \) and \( m'_1, \ldots, m'_n \), and let \( l \) be the length of their maximal common suffix, starting with \( m_{n-(l-1)} = m'_{n'-(l-1)} \). Let us assume (w.l.o.g.) that \( n \geq n' \). It must be true that \( l = n' \), because if \( l < n' \), then by the injectivity of \( M_0 \) (Lemma [28]) there is another common suffix starting with \( m_{n-l} = m'_{n'-l} \), which contradicts our choice of maximal common suffix. Since \( l = n' \), we have that \( m_{n-(l-1)} = m'_{n'-(l-1)} = m'_1 \in M_{Trace}. \) But then \( M_0(m_{n-(l-1)}) \) is undefined, therefore \( m_{n-(l-1)} \) has no predecessors in the sequence \( m_1, \ldots, m_n \), and \( l = n = n' \), and the two sequences are identical.

**Proof of Lemma 33.** We want to prove that

\[
\forall n \in \mathbb{N}_{\geq 1}, \ dist_M^{LTL}(\{n\}^{LTL}_M) = n \  \ (2)
\]
and that
\[ \forall m \in M_{\text{TraceSuffix}}, [\text{dist}^{LTL}_M(m)]^{LTL}_M = m. \] (3)

First, we will prove (2) by induction on \( n \). For the case when \( n = 1 \), we have 
\[ [1]^{LTL}_M \subseteq \llbracket \text{Trace} \rrbracket, \] therefore \( [1]^{LTL}_M \) is the (unique) initial sequence of \([1]^{LTL}_M\), and therefore \( \text{dist}^{LTL}_M([1]^{LTL}_M) = 1 \). For the case when \( n = n' + 1 \), we assume the induction hypothesis \( \text{dist}^{LTL}_M([n']^{LTL}_M) = n' \), from which it follows that \( \text{dist}^{LTL}_M((s_M(n'))) = \{n'\} \). By the definition of \( [\cdot]^{LTL}_M \), we have \( \{n' + 1\} = M_0(t_M(n')) \); this element extends the initial sequence of \([n']^{LTL}_M\) into the initial sequence of \([n' + 1]^{LTL}_M\). Therefore, \( \text{dist}^{LTL}_M([n' + 1]^{LTL}_M) = \text{dist}^{LTL}_M([n']^{LTL}_M) + 1 = n' + 1 \). Second, we will prove (3) by induction on \( \text{dist}^{LTL}_M(m) \). For the case when \( \text{dist}^{LTL}_M(m) = 1 \), by the definition in Lemma 33, we have \( m = M_{\text{Trace}} \), we have \([1]^{LTL}_M = M_{\text{Trace}} \) by definition. For the case when \( \text{dist}^{LTL}_M(m) = n + 1 \), by the definition of \( \text{dist}^{LTL}_M \), there exists some \( m' \in M_{\text{TraceSuffix}} \) such that \( m = M_0(m') \) and \( \text{dist}^{LTL}_M(m') = n \). By the induction hypothesis, \([n]^{LTL}_M = [\text{dist}^{LTL}_M(m')]^{LTL}_M = m' \). Therefore, \( [\text{dist}^{LTL}_M(m)]^{LTL}_M = [n + 1]^{LTL}_M = m'' \) where \( \{m''\} = t_M(n + 1) = M_0(t_M(n)) = M_0(\{m'\}) = M_0(m') \). So we have \( m, m'' \in M_0(m') \), and since \( M_0(\cdot) \) returns a singleton set (Lemma 28), it follows that \( m = m'' \) and therefore 
\[ [\text{dist}^{LTL}_M(m)]^{LTL}_M = m. \]

\( \square \)

Proof of Lemma 3 By structural induction on \( \varphi \).

- For \( \varphi \equiv a \),
  \[
  [L2M'(a)]_{M(\mathcal{T}(M))} = [tr_a]_{M(\mathcal{T}(M))}
  = \{n \in \mathbb{N}_{\geq 1} \mid a \in T(M)[r]\}
  = \{n \in \mathbb{N}_{\geq 1} \mid [n]^{LTL}_M \in tr_a M\}
  = \{\text{dist}^{LTL}_M(m) \mid m \in tr_a M\}
  = \text{dist}^{LTL}_M(tr_a M)
  = \text{dist}^{LTL}_M([tr_a M])
  = \text{dist}^{LTL}_M([L2M'(a)])_M,
  \]
  where the fourth equality holds for the following reason: \( n \in \{n \in \mathbb{N}_{\geq 1} \mid [n]^{LTL}_M \in tr_a M\} \) iff \( [n]^{LTL}_M \in tr_a M \) iff \( n = \text{dist}^{LTL}_M([n]^{LTL}_M) \in \{\text{dist}^{LTL}_M(m) \mid m \in tr_a M\} \), where \( n = \text{dist}^{LTL}_M([n]^{LTL}_M) \) by Lemma 7.
• For $\varphi \equiv \neg \varphi'$,
  \[
  |L2M'(\neg \varphi')|_{\mathcal{M}(T(M))} = \neg |L2M'(\varphi')|_{\mathcal{M}(T(M))} = |L2M'(\varphi')|_{\mathcal{M}(T(M))} = |\mathcal{N}_{\geq 1} \setminus \text{dist}_{\mathcal{LTL}}(L2M'(\varphi')|_{M}) = \text{dist}_{\mathcal{LTL}}(\mathcal{N}_{\geq 1} \setminus |L2M'(\varphi')|_{M}) = \text{dist}_{\mathcal{LTL}}(|L2M'(\varphi')|_{M}) = \text{dist}_{\mathcal{LTL}}(|L2M'(\varphi')|_{M}) = \text{dist}_{\mathcal{LTL}}(|L2M'(\varphi')|_{M}).
  \]

• For $\varphi \equiv \varphi_1 \land \varphi_2$, the result follows by simplifications and using the induction hypothesis.

• For $\varphi \equiv \Diamond \varphi'$,
  \[
  |L2M'(\Diamond \varphi')|_{\mathcal{M}(T(M))} = |\diamond L2M'(\varphi')|_{\mathcal{M}(T(M))} = \text{appext}_{\mathcal{M}(T(M))}(\#\text{next}, |L2M'(\varphi')|_{M(\mathcal{LTL})}) = \text{app}_{\mathcal{M}(T(M))}(\#\text{next}. \text{dist}_{\mathcal{LTL}}(|\text{Trace}| \in L2M'(\varphi')|_{M})) = \text{dist}_{\mathcal{LTL}}(M_{\#\text{next}}(L2M'(\varphi')|_{M})) = \text{dist}_{\mathcal{LTL}}(\text{appext}_{\mathcal{M}(\mathcal{LTL})}(\#\diamond |M|, |L2M'(\varphi')|_{M})) = \text{dist}_{\mathcal{LTL}}(\#(\diamond L2M'(\varphi')|_{M})) = \text{dist}_{\mathcal{LTL}}(|L2M'(\Diamond \varphi')|_{M}),
  \]

where the fourth equality holds for the following reason:

\[
  n \in \text{app}_{\mathcal{M}(\mathcal{LTL})}(\#\text{next}, \text{dist}_{\mathcal{LTL}}(|\text{Trace}| \in L2M'(\varphi')|_{M})) \iff n + 1 \in \text{dist}_{\mathcal{LTL}}(|\text{Trace}| \in L2M'(\varphi')|_{M}) \iff \exists m \in |\text{Trace}| \in L2M'(\varphi')|_{M}. n + 1 = \text{dist}_{\mathcal{LTL}}(m) \iff \exists m \in |\text{Trace}| \in L2M'(\varphi')|_{M}. n + 1 = \text{dist}_{\mathcal{LTL}}(m) \iff \exists m \in |\text{Trace}| \in L2M'(\varphi')|_{M}. n + 1 = \text{dist}_{\mathcal{LTL}}(m) \iff \exists m \in |\text{Trace}| \in L2M'(\varphi')|_{M}. n = \text{dist}_{\mathcal{LTL}}(m) \iff n = \text{dist}_{\mathcal{LTL}}(M_{\#\text{next}}(L2M'(\varphi')|_{M})).
\]
• For $\varphi \equiv \varphi_1 U \varphi_2$, by definition of the $U$ operator, we have
\[
\begin{align*}
\text{dist}^{\text{LTL}}_M([L2M'((\varphi_1 U \varphi_2))_M]) \\
= \text{dist}^{\text{LTL}}_M([\mu X . \varphi_2 \lor (\varphi_1 \land oX)]_M) \\
= \text{dist}^{\text{LTL}}_M(\mu F),
\end{align*}
\]
where
\[
F(X) = [\varphi_2]_M \cup ([\varphi_1]_M \cap M_0(X)).
\]
We also have
\[
\begin{align*}
[L2M'((\varphi_1 U \varphi_2))_{M(\mathcal{T}(M))}] \\
= [\mu X . \varphi_2 \lor (\varphi_1 \land oX)]_{M(\mathcal{T}(M))} \\
= \mu G,
\end{align*}
\]
where
\[
G(X) = [\varphi_2]_{M(\mathcal{T}(M))} \\
\quad \cup ([\varphi_1]_{M(\mathcal{T}(M))} \cap M(\mathcal{T}(M))_o(X)) \\
= \text{dist}^{\text{LTL}}_M([\varphi_2]_M) \cup \\
(\text{dist}^{\text{LTL}}_M([\varphi_1]_M) \cap \text{dist}^{\text{LTL}}_M(M_0([X]_M)))) \\
= \text{dist}^{\text{LTL}}_M([\varphi_2]_M \cup ([\varphi_1]_M \cap M_0([X]_M))),
\]
where the second equality follows from the induction hypothesis. Now we need to show that $\text{dist}^{\text{LTL}}_M(\mu F) = \mu G$. First, $\text{dist}^{\text{LTL}}_M(\mu F)$ is a fixpoint of $G$:
\[
G(\text{dist}^{\text{LTL}}_M(\mu F)) = \text{dist}^{\text{LTL}}_M([\varphi_2]_M \cup ([\varphi_1]_M \cap M_0(\mu F))) \\
= \text{dist}^{\text{LTL}}_M(\mu F).
\]
It is also the least fixpoint. Let $A$ be a fixpoint of $G$. Then $[A]_M$ is a fixpoint of $F$ by
\[
[A]_M = [G(A)]_M
= [\varphi_2]_M \cup ([\varphi_1]_M \cap M_0([A]_M))
= F([A]_M)
\]
and from $\mu F$ being the least fixpoint of $F$ it follows that $\mu F \subseteq [A]_M$, and therefore $[\mu F]_M \subseteq A$.

Proof of Lemma 3. $\mathcal{T}(M)$, $i \models_{\text{LTL}} \varphi$ iff (using Lemma 3) $i \in [L2M'((\varphi))_{M(\mathcal{T}(M))}]$ iff (using Lemma 7 and Lemma 8) $\text{dist}^{\text{LTL}}_M([i]_M) \in \text{dist}^{\text{LTL}}_M([L2M'((\varphi))]_M)$ iff (since $\text{dist}^{\text{LTL}}_M$ is injective by Lemma 7) $[i]_M \in [L2M'((\varphi))]_M$. □

Proof of Theorem 10. The proof goes similarly to the proof of Theorem 4 except that it uses Lemma 9 instead of Lemma 3. □
A.3 Proofs About LTL Model Checking in Matching Logic

Definition 34. Given a $\Gamma^{KS}$-model $M$, we define a Kripke structure $\text{KSof}(M) = (S, S_0, \delta, L)$ as follows:

- $S = \text{appext}_M(\text{inh}_M, \text{State}_M)$;
- $S_0 = \text{appext}_M(\text{inh}_M, \text{InitState}_M)$;
- $\delta(s) = \text{appext}_M((\_ \rightarrow \_), \{s\})$;
- $L(s) = \{a \in \text{AP} \mid s \in (st_a)_M\}$.

Definition 35. Given a Kripke structure $K = (S, S_0, \delta, L)$, we define a $\Sigma^{KS}$-model $M = \text{MSKS}(K)$ as follows.

- The carrier set $M = S \uplus \{\#\text{def}, \#\text{inh}, \#\text{State}, \#\text{InitState}, \#\text{trans}\}$.

- The interpretation of symbols:
  - $\text{def}_M = \{\#\text{def}\}$;
  - $\text{inh}_M = \{\#\text{inh}\}$;
  - $\text{State}_M = \{\#\text{State}\}$;
  - $\text{InitState}_M = \{\#\text{InitState}\}$;
  - $(\_ \rightarrow \_)_M = \{\#\text{trans}\}$;
  - $(st_a)_M = \{s \mid a \in L(s)\}$ for any $a \in \text{AP}$.

- The interpretation of application:
  - $\text{app}_M(\#\text{def}, m) = M$ for any $m \in M$;
  - $\text{app}_M(\#\text{inh}, \#\text{State}) = S$;
  - $\text{app}_M(\#\text{inh}, \#\text{InitState}) = S_0$;
  - $\text{app}_M(\#\text{trans}, s) = \delta(s)$ for any $s \in S$;
  - $\text{app}_M(m_1, m_2) = \emptyset$ otherwise.

Lemma 36. $\text{MSKS}(K) \models \Gamma^{KS}$ for any Kripke structure $K$

Proof of lemma 36. Straightforward verification of the axioms.

We define the notion of isomorphism of matching logic models and prove that isomorphic models satisfy the same formulas.

Definition 37. Given a matching logic signature $\Sigma$ and two matching logic $\Sigma$-models $(M_1, \{\sigma_{M_1}\}_{\sigma \in \Sigma}, \text{app}_{M_1})$ and $(M_2, \{\sigma_{M_2}\}_{\sigma \in \Sigma}, \text{app}_{M_2})$, and isomorphism between $M_1$ and $M_2$ is a bijective function $f : M_1 \rightarrow M_2$ such that:

- $f(\sigma_{M_1}) = \sigma_{M_2}$ for any $\sigma \in \Sigma$; and
- $f(\text{app}_{M_1}(m, n)) = \text{app}_{M_2}(f(m), f(n))$ for any $m, n \in M_1$.

We say that $M_1$ and $M_2$ are isomorphic (and write $M_1 \cong M_2$) iff there is an isomorphism between them.
Lemma 38 (Isomorphism preserves semantics). Given an isomorphism $f$ between $M_1$ and $M_2$, an $M_1$-valuation $\rho$, and a pattern $\varphi$,

$$f(|\varphi|_{M_1,\rho}) = |\varphi|_{M_2,f \circ \rho},$$

where $(f \circ \rho)(x) = f(\rho(x))$ for any element variable $x$, and $(f \circ \rho)(X) = \{f(m) | m \in \rho(X)\}$ for any set variable $X$.

Proof. We omit the proof from the text, as it is proven in [1], file ModelIsomorphism.v, lemma isomorphism_preserves_semantics.

Lemma 39. Isomorphic Kripke structures translate to isomorphic matching logic models:

$$K_1 \simeq K_2 \Rightarrow M_{K_S}(K_1) \simeq M_{K_S}(K_2).$$

Proof. We simply extend the isomorphism of Kripke structures.

Lemma 40. $K_Sof(M_{K_S}(K)) = K$ for any Kripke structure $K$.

Proof. Straightforward.

Proof of Theorem 17. The proof is by induction on the size of the formulas, using Definition 15 and the semantics of the individual constructs. We do not show it here in text, because it is straightforward but tedious, and depends on lots of lemmas about the individual constructs (e.g., about equality, subseteq etc.). A mechanized version of the proof can be found in [1], file ModelExtension.v, lemma semantics_preservation.

A.4 A Linking Theory

We want to prove that $\Gamma^{\text{LINK}}$ correctly links LTL models (traces) and Kripke structures. First, we generalize Definition 5.

Definition 41. For any $\Gamma^{\text{LTL}}$-model $M$, we define the function $[\cdot]_{M} : M_{\text{TraceSuffix}} \times \mathbb{N}_{\geq 1} \to M_{\text{TraceSuffix}}$ by

$$m[n..\infty]_M = \begin{cases} m & \text{if } n = 1 \\ M_{\text{suffix}}(m[(n-1)\ldots\infty]_{M}^{\text{LTL}}) & \text{if } n > 1 \end{cases}$$

Directly from the definition, it follows that $[n]^{\text{LTL}}_M = M_{\text{Trace}}[n..\infty]_M$. Similarly, we generalize Definition 6.

Definition 42. For any $\Gamma^{\text{LTL}}$-model $M$, let $\tau_M : M_{\text{TraceSuffix}} \to (\mathcal{P}(\text{AP}))^\omega$ be a function defined by

$$\tau_M(m)(i) = \{a \in \text{AP} | m[i..\infty]_M \in (st_a)_M\}.$$

It follows that $\mathcal{T}(M) = \tau_M(M_{\text{Trace}})$. 32
Spec LINKPRELIM
Import: KS, LTL
Symbols: PairStateTraceSuffix, psts, πst, πtr
Axioms:

(LTLGLPair1) πst : PairStateTraceSuffix → State
(LTLGLPair2) πtr : PairStateTraceSuffix → TraceSuffix
(LTLGLPair)
\[ \forall st : State. \forall tr : TraceSuffix. \exists p : PairStateTraceSuffix. (psts st tr) = p \]
(LTLGLPinj)
\[ \forall st_1, st_2 : State. \forall tr_1, tr_2 : TraceSuffix. (psts st_1 tr_1) = (psts st_2 tr_2) \rightarrow st_1 = st_2 \land tr_1 = tr_2 \]
(LTLGLi0)
\[ \forall y : PairStateTraceSuffix. (psts (πst y) (πtr y)) = y \]
(LTLGLi1)
\[ \forall st : State. \forall tr : TraceSuffix. (πst (psts st tr)) = st \]
(LTLGLi2)
\[ \forall st : State. \forall tr : TraceSuffix. (πtr (psts st tr)) = tr \]

Spec. 6: A theory gluing Kripke structures and LTL reasoning - part 1

**Definition 43.** Let \( M \) be a \( (\Gamma_{LTL} \cup \Gamma_{KS} \cup \Gamma_{LINKPRELIM}) \)-model, let \( T = M_{TraceSuffix} \), and let \( (S, \cdot, \delta, L) = KSof(M) \). We define a binary relation \(~M_{AP}\subseteq S \times T\) by
\[ s ~M_{AP} t \iff L(s) = \tau_M(t)(1). \]
Furthermore, we define a ternary relation \( (\sim_M) ~M_{AP} \subseteq S \times T \times P(M) \) by
\[ (s, t) ~M_{AP} A \iff \text{there exists some } m' \in M \text{ and } s' \in S \text{ such that } s' \in \delta(s) \text{ and } psts_M \circ_{M} \{s'\} \circ_{M} \{M_0(t)\} = \{m'\} \text{ and } m' \in A \]

**Lemma 44.** Let \( M \) be a \( (\Gamma_{LTL} \cup \Gamma_{KS} \cup \Gamma_{LINKPRELIM}) \)-model, let \( T = M_{TraceSuffix} \), and let \( (S, \cdot, \delta, L) = KSof(M) \). For any two element variables \( x, y \), for any \( s \in S \) and any \( t \in T \), and for any \( M \)-valuation \( \rho \), we have
\[ s ~M_{AP} t \iff \left| x \sim_{AP} y \right|_{M, \rho}|x \sim s|_{\rho} \cdot |y \sim t| \]
furthermore, for any set $A \subseteq M$ and any set variable $X$,

$$(s, t) \sim_M^3 A \iff [(x, y) \sim_M^3 X]_{M, \rho[x\mapsto s][y\mapsto t][X\mapsto A]}$$

where $\sim_{AP}$ and $\sim_M^3$ are as defined in Spec. 5.

Proof. For the first property, we have

$s \sim_{AP}^M t$

iff (by Definition 43)

$L(s) = \tau_M(t)(1)$

iff (by Definition 42)

$L(s) = \{a \in AP \mid m[1..\infty]_M \in (tr_a)_M\}$

iff

$L(s) = \{a \in AP \mid m \in (tr_a)_M\}$

iff (by Definition 34)

$\{a \in AP \mid s \in (st_a)_M\} = \{a \in AP \mid m \in (tr_a)_M\}$

iff

$s \in (st_a)_M \iff m \in (tr_a)_M$ (4)

For the second property, we have

$(s, t) \sim_M^3 A$

iff (by Definition 43)

there exists some $m' \in M$ and $s' \in S$ such that $s' \in \delta(s)$ and

$psts_M @_M \{s'\} @_M \{M_0(t)\} = \{m'\}$ and $m' \in A$

iff (by Definition 34 and some standard matching logic reasoning)

$[(x, y) \sim_{AP} y]_{M, \rho[x\mapsto s][y\mapsto t]}$

Now we give a semantic characterization of the relational pattern $Corr$.

Definition 45.

$M_{Corr} = \{ m \mid \exists st \in S. \exists tr \in T. m \in psts_M @_M \{st\} @_M \{tr\} \land
\exists st'_0, st'_1, st'_2, \ldots \in S$ such that

$st'_0 = st \land \forall i \geq 0. s_{i+1} \in \delta(s_i) \land \tau_M(tr)(i) = L(st'_i),$

where $(S, \delta, \rho, L) = KSof(M)\}$
Lemma 46 (Semantics of Corr). Let $M$ be a $(\Gamma^{\text{LTL}} \cup \Gamma^{\text{KS}} \cup \Gamma^{\text{LINKPRELIM}})$-model, let $T = M_{\text{TraceSuffix}}$, and let $(S, \cdot, \delta, L) = \text{KSoF}(M)$. Then for any $M$-valuation $\rho$, 

$$\eval{\text{Corr}}{M, \rho} = M_{\text{Corr}}$$

Proof. Using the semantics for greatest fixpoint and Lemma 44, we have to show that $M_{\text{Corr}} = \nu F$, where $F : \mathcal{P}(A) \to \mathcal{P}(A)$ is defined by

$$F(A) = \{m \mid \exists s \in S, t \in T \text{ s.t. } m \in \text{psts}_M \otimes_M \{s\} \otimes_M \{t\} \text{ and } s \sim^M A, t \text{ and } (s, t) \sim^3_M A\}$$

By the Knaster-Tarski fixpoint theorem, it is enough to show that $M_{\text{Corr}}$ is the greatest fixpoint of $F$. Let $(S, \cdot, \delta, L) = \text{KSoF}(M)$.

- First, we show that $M_{\text{Corr}} \subseteq F(M_{\text{Corr}})$. Let $m \in M_{\text{Corr}}$. By definition of $M_{\text{Corr}}$, let $s=t \in S$ and $tr \in T$ such that $m \in \text{psts}_M \otimes_M \{st\} \otimes_M \{tr\}$, and let $st'_0, st'_1, \ldots \in S$ such that $st'_0 = s t$ and $(H) \forall i \geq 0, s_{i+1} \in \delta(s_i) \wedge \tau_M(tr)(i) = L(st'_i)$. To show that $m \in F(M_{\text{Corr}})$, we have to show that $st'_0 \sim^M tr$ and $(st'_0, tr) \sim^3_M M_{\text{Corr}}$. The first holds easily by definition of $\sim^M_A$ and choice $i = 1$ in $H$. For the second we unfold the definition, choose $i = 1$ in $H$ and have to prove that the (unique) $m'$ such that $(m') = \text{psts}_M \otimes_M \{st'_0\} \otimes_M \{M_0(tr)\}$ satisfies $m' \in M_{\text{Corr}}$. But that is true, since we can choose the sequence $st'_1, st'_2, \ldots$ and specialize $H$ to $i + 1$.

- Now we want to prove that $F(M_{\text{Corr}}) \subseteq M_{\text{Corr}}$. This holds by a similar (but opposite) argument as the previous point: given some $m \in F(M_{\text{Corr}})$, it follows that there exists some $m' \in M_{\text{Corr}}$, and we prove that $m \in M_{\text{Corr}}$ by extending the sequence $st'_0, st'_1, \ldots$ from left with the state-projection of $m$.

- It remains to prove that $M_{\text{Corr}}$ is the greatest among all the fixpoints of $F$. Let $B$ be some fixpoint of $F$. We prove that $B \subseteq M_{\text{Corr}}$ as follows. Since $B = F(B) = F^2(B) = \ldots$, it follows that for any $m \in B$ we have an infinite sequence $(s_0, t_0), (s_1, t_1), \ldots \in S \times T$ such that $m \in \text{psts}_M \otimes_M \{s_0\} \otimes_M \{t_0\}$ and for any $i \geq 0, s_{i+1} \in \delta(s_i)$ and $t_{i+1} = M(t_i)$ and $s_i \sim^M_A t_i$. But then $m \in M_{\text{Corr}}$. 

Proof of Lemma 24. We have 

$$\text{Trace}(\text{KSoF}(M))$$

iff, by Definition 42

$$\tau_M(M_{\text{Trace}}) \in \text{Trace}(\text{KSoF}(M))$$

iff, by Definition 13

$$\exists s_0, s_1, \ldots \in S \text{ such that } \forall i \geq 0, s_{i+1} \in \delta(s_i) \text{ and } \tau_M(M_{\text{Trace}})(i) = L(s_i) \text{ and } s_0 \in S_0$$

(where $(S, S_0, \delta, L) = \text{KSoF}(M)$) iff, by Definition 45

$$\exists s_0 \in S_0, \exists m \in M. \{m\} = \text{apext}_M(\text{apext}_M(\text{psts}_M, \{s_0\}), \{M_{\text{Trace}}\}) \text{ and } m \in M_{\text{Corr}}$$
iff, by Lemma [46] for any $M$-valuation $\rho$,
\[ \exists s_0 \in S_0. \exists m \in M. \{ m \} = \text{appext}_M(\text{appext}_M(psts_M, \{ s_0 \}), \{ M_{\text{Trace}} \}) \text{ and } m \in Corr_M, \rho \]
iff, by Definition [34] and some standard matching logic reasoning, for any $M$-valuation $\rho$,
\[ \exists s_0 : \text{InitState}. \langle s_0, \text{Trace} \rangle \in Corr_M, \rho \]
iff
\[ M \models \Gamma_{\text{LINK}}. \]
\[ \square \]

A.5 Model Checking

We let $M_{\text{State}} = \text{appext}_M(\text{inh}_M, \text{State}_M)$ for any $\Sigma^{KS}$-model $M$.

We define a model extension that adds pairs to a $\Sigma^{KS} \cup \Sigma^{LTL}$ model.

**Definition 47.** Let $M$ be a $\Sigma^{KS} \cup \Sigma^{LTL}$-model. We define an extended model $\text{addPairs}(M)$ satisfying $\Gamma_{\text{LINKPRELIM}}$. By Definition [16] the model isomorphism theorem (Lemma [38]), and the fact that we can extend model along signature inclusion without changing semantics of formulas from the original language, the extended model $\text{addPairs}(M)$ satisfies the same open predicates over $\Sigma^{MC} \cup \Sigma^{LTL}$ as $M$. The construction is as follows. Let $P = M_{\text{State}} \times M_{\text{TraceSuffix}}$. We let $\#\text{def}, \#\text{inh}, \#\text{projst}, \#\text{projtr}, \#\text{pair}, \#\text{pairsort}$ be fresh elements. Without loss of generality, we assume that $M \cap P = \emptyset$ (if not, we rename elements of $M$ and apply the model isomorphism theorem). We define $\text{addPairs}(M) = (M', \{ \sigma_{M'} \}_{\sigma \in \Sigma^{\text{LINKPRELIM}}}, \text{app}_{M'}(\cdot, \cdot))$ by

- defining the carrier set $M' = M \cup P \cup \{ \#\text{def}, \#\text{inh}, \#\text{projst}, \#\text{projtr}, \#\text{pair}, \#\text{pairsort} \} \cup (\{\#\text{pair}\} \times M_{\text{State}})$;
- defining the interpretation of symbols by
  - $\sigma_{M'} = \sigma_M$ for any $(\sigma \in \Sigma^{MC} \cup \Sigma^{LTL}) \setminus \{ \text{def, inh} \}$;
  - $\text{psts}_{M'} = \{ \#\text{pairsort} \}$;
  - $\pi_{st} = \{ \#\text{projst} \}$; and
  - $\pi_{tr} = \{ \#\text{projtr} \}$; and
- defining the application by
  - $\text{app}_{M'}(\#\text{def}, m) = M'$ for any $m \in M'$;
  - $\text{app}_{M'}(\#\text{inh}, m) = \text{appext}_M(\text{inh}_M, \{ m \})$ for any $m \in M$;
  - $\text{app}_{M'}(\#\text{inh}, \#\text{pairsort}) = P$;
  - $\text{app}_{M'}(\#\text{projst}, (st, tr)) = \{ st \}$ for any $st \in M_{\text{State}}$ and $tr \in M_{\text{TraceSuffix}}$;
  - $\text{app}_{M'}(\#\text{projtr}, (st, tr)) = \{ tr \}$ for any $st \in M_{\text{State}}$ and $tr \in M_{\text{TraceSuffix}}$;
  - $\text{app}_{M'}(\#\text{pair}, st) = (\#\text{pair}, st)$ for any $st \in M_{\text{State}}$;
  - $\text{app}_{M'}((\#\text{pair}, st), (st, tr)) = (st, tr)$ for any $st \in M_{\text{State}}$ and $tr \in M_{\text{TraceSuffix}}$.

It is easy to verify that $M' \models \Gamma_{\text{LINKPRELIM}}$ whenever $M \models \Gamma^{KS} \cup \Gamma^{LTL}$.
Lemma 48. Model extension preserves $KSof(\cdot)$. 

Proof. We observe that $S, S_0, \delta(s)$, and $L(s)$ from Definition 34 can be alternatively defined by querying the model $M$ with open formulas:

- $S = \text{appext}_M(\text{inh}_M, \text{State}_M) = |\text{State}_M|_M$;
- $S_0 = \text{appext}_M(\text{inh}_M, \text{InitState}_M) = |\text{InitState}_M|_M$;
- $\delta(s) = \text{appext}_M(\langle \_ \rightarrow \_ \rangle, \{s\}) = |\langle \_ \rightarrow \_ \rangle x|_{M, \rho}$ where $\rho(x) = s$;
- $a \in L(s) \iff |x \in st_a|_{M, \rho} = M$ where $\rho(x) = s$.

Since these open formulas are preserved by model extension, they are also preserved by model gluing and the pairs extension. 

Lemma 49. Model extension preserves $T$. 

Proof. By the same argument as the previous lemma: $a \in T(M)(i) \iff |(\delta^t \cdot T) \subseteq tr_a|_M = M$. 

Proof of Theorem 21. Assume that $M \models^S \iff KSof(M) \cong S$ for any matching model $M$. We have 

$S \models_{\text{LTL}} \varphi$ 

iff (by Definition 14) 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}. \tau \in \text{Traces}(S) \implies \tau \models_{\text{LTL}} \varphi$ 

iff (by Theorem 4) 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}. \tau \in \text{Traces}(S) \implies M(\tau) \models \varphi_{\text{Trace}}$ 

iff (by correctness of model gluing) 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}. \tau \in \text{Traces}(S) \implies \text{glue}(M(\tau), \mathcal{M}_{KS}(M)) \models \varphi_{\text{Trace}}$ 

iff (by Definition 47) 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}. \tau \in \text{Traces}(S) \implies \text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(M)))) \models \varphi_{\text{Trace}}$. 

Thanks to Lemmas 48 and 49 we have 

- $T(M(\tau)) = T(\text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(M))))$ and 
- $S = KSof(M_{KS}(S)) = KSof(\text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(M))))$;

therefore, the above holds iff 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}.

T(\text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(S))))) \in \text{Traces}(KSof(\text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(S)))))) 

\implies \text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(S)))) \models \varphi_{\text{Trace}}$ 

iff (by Lemma 20, since $\text{addPairs}((\ldots)) \models \Gamma_{\text{LINKPRELIM}}$) 

$\forall \tau \in (\mathcal{P}(\mathcal{AP}))^{\omega}.

\text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(S)))) \models \varphi_{\text{LINK}} 

\implies \text{addPairs}((\text{glue}(M(\tau), \mathcal{M}_{KS}(S)))) \models \varphi_{\text{Trace}}$ 

37
iff
\[ \forall \tau \in (\mathcal{P}(\mathcal{AP}))^\omega, \forall M_S \in \text{Mod}_{\text{ML}}(\Gamma^{K_S}), M_S \models \Gamma^S \implies \] 
\[ \text{addPairs}(\text{glue}(M(\tau), M_S)) \models \Gamma^{\text{LINK}} \] 
\[ \text{addPairs}(\text{glue}(M(\tau), M_S)) \models \models_{\text{Trace}} \]

(the bottom to top implication is straightforward, since \( M_{K_S}(S) \models \Gamma^S \); for the other implication we use the fact that \( M_{K_S}(S) \cong M_{K_S}(K\text{Sof}(M_S)) = M_S \) and that isomorphic matching logic model satisfy the same formulas - Lemma [35]. This is then equivalent to
\[ \forall M_\tau \in \text{Mod}_{\text{ML}}(\Sigma^{\text{LTL}}), M_\tau \models \Gamma^{\text{LTL}} \implies \] 
\[ \forall M_S \in \text{Mod}_{\text{ML}}(\Gamma^{K_S}), M_S \models \Gamma^{K} \implies \] 
\[ \text{addPairs}(\text{glue}(M_\tau, M_S)) \models \Gamma^{\text{LINK}} \] 
\[ \text{addPairs}(\text{glue}(M_\tau, M_S)) \models \models_{\text{Trace}} \]

where the bottom to top implication is easy. For the other implication, we use the fact that \( M_\tau \) and \( M(\ell(M_\tau)) \) satisfy the same set of LTL formulas (a consequence of Lemma [35], change the conclusion of the goal to \( \text{addPairs}(\text{glue}(M(\ell(M_\tau)), M_S)) \models \models_{\text{Trace}} \), apply the hypothesis from top, and are left to prove \( \text{addPairs}(\text{glue}(M(\ell(M_\tau)), M_S)) \models \Gamma^{\text{LINK}} \) from \( \text{addPairs}(\text{glue}(M_\tau, M_S)) \models \Gamma^{\text{LINK}} \), which we prove by double application of Lemma [20] using the fact that \( \ell(M(\tau)) = \tau \) and that \( \text{addPairs}(\text{glue}(\_, \_)) \) preserves \( \ell \). The above is then equivalent to
\[ \Gamma^S \cup \Gamma^{MC} \models \models_{\text{Trace}} \]

where the bottom to top implication follows from correctness of model gluing (since \( \Gamma^S \cup \Gamma^{MC} \) contains only open predicates). The top to bottom implication is proved by contraposition as follows. Assume there is some model \( M \models \Gamma^S \cup \Gamma^{MC} \) such that \( M \notmodels \models_{\text{Trace}} \). We need to translate this model \( M \) into some \( M_\tau \in \text{Mod}_{\text{ML}}(\Gamma^{LTC}) \) and \( M_S \in \text{Mod}_{\text{ML}}(\Gamma^{K}) \) such that \( \text{addPairs}(\text{glue}(M_\tau, M_S)) \models \Gamma^{\text{LINK}} \) and \( \text{addPairs}(\text{glue}(M_\tau, M_S)) \notmodels \models_{\text{Trace}} \); that is, we need to split the glued counterexample back into a model of Kripke structure and a model of a trace. We let \( M_\tau = M(\ell(M)) \) and \( M_S = M_{K_S}(K\text{Sof}(M)) \), and have to prove
\[ \text{addPairs}(\text{glue}(M(\ell(M)), M_{K_S}(K\text{Sof}(M)))) \models \Gamma^{\text{LINK}} \]
and
\[ \text{addPairs}(\text{glue}(M(\ell(M)), M_{K_S}(K\text{Sof}(M)))) \notmodels \models_{\text{Trace}} \cdot \]
The second obligation holds because we can strip the \( \text{addPairs} \) part and then apply the correctness of gluing. To prove the first obligation, first apply correctness of \( \Gamma^{\text{LINK}} \) (Lemma [20]) to get
\[ \ell(\text{addPairs}(\text{glue}(M(\ell(M)), M_{K_S}(K\text{Sof}(M)))))) \] 
\[ \in \text{Traces}(K\text{Sof}(\text{addPairs}(\text{glue}(M(\ell(M)), M_{K_S}(K\text{Sof}(M)))))) \cdot \]
then note that \( \text{addPairs}(\text{glue}(\_, \_)) \) preserves \( \ell \) and KSoF to get
\[ \ell(M(\ell(M))) \] 
\[ \in \text{Traces}(K\text{Sof}(M_{K_S}(K\text{Sof}(M)))) \cdot \]

38
then simplify the goal to
\[ T(M) \in Traces(KSof(M)). \]
and apply the correctness of gluing again, getting
\[ M \models \Gamma^\text{LINK}, \]
which follows from the assumption that \( M \models \Gamma^S \cup \Gamma^M \). This concludes the proof. \qed