Contents

Chapter 12

Linear Temporal Logic

12.1 Linear Temporal Logic

In the last chapter, we studied how to use ω-automata to specify liveness requirements regarding infinite behaviors of fair modules. Such requirements can be alternatively, and more succinctly, specified using the temporal logic.

12.1.1 Syntax and Semantics

While the formulas of branching-time logics such as CTL are interpreted over trees, the formulas of *linear temporal logic* (LTL) are interpreted over ω-words over observations. As in case of CTL, the logic LTL employs temporal modalities such as *next*, *always*, *eventually*, and *until*. While in CTL every temporal connective has two types; existential and universal (e.g. possibly-next $\exists\bigcirc$ vs. inevitably-next $\forall\bigcirc$), such a distinction is not necessary in LTL whose formulas are interpreted over a fixed ω-word. On the other hand, while LTL admits nesting of temporal connectives freely, operators such $\Box\Diamond$ are not expressible in CTL.

> LINEAR TEMPORAL LOGIC: SYNTAX
>
> The formulas of *Linear Temporal Logic* (LTL) are defined inductively by the grammar
>
> $$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\varphi$$
>
> for observation predicates p.

The LTL formulas are interpreted over the positions of an infinite sequence of observations. Consider an ω-word $\underline{a}$ over the alphabet A whose symbols give interpretation to the predicates appearing in an LTL formula. The truth of an atomic predicate at the position i of $\underline{a}$ is evaluated according to the corresponding observation a_i of $\underline{a}$. The *next-formula* $\bigcirc\varphi$ holds at the position i of $\underline{a}$ iff the formula φ holds at the next-position $i+1$ of $\underline{a}$. The *until-formula* $\varphi_1\mathcal{U}\varphi_2$ holds at the position i of $\underline{a}$ iff theres exists a later position $j \geq i$ such that the formula φ_2 holds at the position j and the formula φ_1 holds at all positions k such that $i \leq k < j$.

For an LTL formula φ, the set A_φ contains the set of all possible valuations to the atomic predicates appearing in φ.

LINEAR TEMPORAL LOGIC: SEMANTICS

An LTL formula φ is interpreted at the positions of ω-words over the set A_φ of observations. For all ω-words $\underline{a}$ and all $i \geq 0$,

$$
\begin{array}{lll}
i \models_{\underline{a}} p & \text{iff} & a_i \models p; \\
i \models_{\underline{a}} \varphi_1 \vee \varphi_2 & \text{iff} & i \models_{\underline{a}} \varphi_1 \text{ or } i \models_{\underline{a}} \varphi_2; \\
i \models_{\underline{a}} \neg\varphi & \text{iff} & i \not\models_{\underline{a}} \varphi; \\
i \models_{\underline{a}} \bigcirc\varphi & \text{iff} & i+1 \models_{\underline{a}} \varphi; \\
i \models_{\underline{a}} \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \text{there is a natural number } j \geq i \text{ such that} \\
 & & j \models_{\underline{a}} \varphi_2 \text{ and for all } i \leq k < j,\ k \models_{\underline{a}} \varphi_1.
\end{array}
$$

The ω-word $\underline{a}$ ***satisfies*** the LTL formula φ, written $\underline{a} \models \varphi$, if $0 \models_{\underline{a}} \varphi$. The ***$\omega$-language*** $\mathcal{L}_\varphi$ defined by φ is the set $\{\underline{a} \in A_\varphi^\omega \mid \underline{a} \models \varphi\}$ of ω-words that satisfy φ. The LTL formula φ is ***satisfiable*** if the ω-language $\mathcal{L}_\varphi$ is nonempty, and ***valid*** if the ω-language $\mathcal{L}_\varphi$ equals A_φ^ω.

The following temporal operators are defined in LTL:

$$
\begin{array}{lll}
\Diamond\varphi & \text{for} & \mathit{true}\,\mathcal{U}\,\varphi; \\
\Box\varphi & \text{for} & \neg\Diamond\neg\varphi; \\
\varphi_1 \mathcal{W} \varphi_2 & \text{for} & \varphi_1 \mathcal{U} \varphi_2 \vee \Box\varphi_1.
\end{array}
$$

The temporal operators $\bigcirc$, $\Box$, $\Diamond$, $\mathcal{U}$, and $\mathcal{W}$ are called *next*, *always*, *eventually*, *until*, and *wait-for*, respectively. The eventually-formula $\Diamond\varphi$ holds at the position i of an ω-word $\underline{a}$ if the formula φ holds at some position $j \geq i$ of $\underline{a}$; the always-formula $\Box\varphi$ holds at the position i of an ω-word $\underline{a}$ if the formula φ holds at all positions $j \geq i$ of $\underline{a}$.

Remark 12.1 [Propositional LTL] The LTL formula φ is *propositional* if every atomic predicate in φ is a propositional formula. For a propositional LTL formula φ, an observation in A_φ is a valuation for the set X_φ of boolean variables appearing in φ, and the set A_φ equals the power-set $\mathbf{2}^{X_\varphi}$. ■

Example 12.1 []LTL languages The LTL formula $\Box p$ defines the safety language containing ω-words all of whose observations satisfy p. The LTL formula $\Diamond p$ defines the guarantee language containing ω-words that contain an observation satisfying p. The LTL formula $\Box\Diamond p$ defines the response language containing ω-words that contain infinitely many observations satisfying p. The LTL formula $\Diamond\Box p$ defines the persistence language containing ω-words with a suffix with only observations satisfying p. Thus, the LTL operators $\Box$, $\Diamond$, $\Box\Diamond$, and $\Diamond\Box$ correspond to the operators *safe*, *guar*, *recur*, and *persist*, respectively. ■

12.1.2 LTL as a specification logic

We can view LTL as a fair state logic, by interpreting LTL formulas over the states of a given fair structure $\mathcal{K}$.

> LTL SEMANTICS OVER FAIR STRUCTURES
> Let φ be an LTL formula, and let $\mathcal{K}$ be a fair structure with observations A_φ. For a state s of $\mathcal{K}$, $s \models_{\mathcal{K}} \varphi$ if all source-s fair ω-traces of $\mathcal{K}$ satisfy φ. The fair structure $\mathcal{K}$ *satisfies* the LTL formula φ if $s \models_{\mathcal{K}} \varphi$ for all initial states s of $\mathcal{K}$.

Thus, a fair structure $\mathcal{K}$ satisfies an LTL formula φ if every fair trace of $\mathcal{K}$ satisfies φ: $\mathcal{L}_{\mathcal{K}} \subseteq \mathcal{L}_\varphi$. The model checking problem and verification problem for LTL are defined as in other logics.

Remark 12.2 [] Let φ be an LTL formula whose atomic predicates are boolean expressions over the set X_φ of variables. Then, the set A_φ is the set Σ_{X_φ} of valuations for X_φ. We can interpret the formula φ over a fair structure whose observations are valuations for a superset of the variables X_φ. ■

Example 12.2 [LTL specifications for mutual exclusion] Recall the mutual exclusion problem from Chapter 2. The mutual exclusion requirement is specified by the LTL formula

$$\varphi_{me}: \ \Box\neg(pc_1 = inC \wedge pc_2 = inC).$$

The first-request-first-in requirement that if process P_1 requests an entry to the critical section while process P_2 is outside, then P_2 cannot overtake P_1 to enter the critical section, is expressed by the formula

$$\varphi_{\mathit{fifo}}: \ \Box\,[\,(pc_1 = reqC \wedge pc_2 = outC) \rightarrow (pc_2 \neq inC)\,\mathcal{W}\,(pc_1 = inC)\,].$$

Finally, the starvation freedom requirement for process P_1 is specified by the formula

$$\varphi_{sf}: \ \Box\,(\,pc_1 = reqC \rightarrow \Diamond pc_1 = inC).$$

■

Remark 12.3 [] The observation predicate p is an invariant of the module P iff the module P satisfies the LTL formula $\Box p$; the observation predicate p is a recurrent of the fair module $\mathcal{P}$ iff the fair module $\mathcal{P}$ satisfies the LTL formula $\Box\Diamond p$; and the observation predicate p is a response to the observation predicate q for the fair module $\mathcal{P}$ iff the fair module $\mathcal{P}$ satisfies the LTL formula $\Box(p \rightarrow \Diamond q)$.
■

Example 12.3 [Producer-consumer requirements] Recall the message-passing protocols from Section 2.3.3, and their fair versions from Section 12.5.3. Let us consider the requirement that if the producer produces a message, say with value m, then eventually the consumer consumes a message with value m. Recall that the producer signals the production of the message by issuing the event $done_P$, and the produced message appears in the variable msg_P. The consumer signals the consumption of the message by issuing the event $done_C$, and the consumed message appears in the variable msg_C. Let $done_P?$ denote the LTL formula $(done_P \leftrightarrow \bigcirc\neg done_P)$, and for a message value m, let $done_P?m$ denote the LTL formula $(done_P \leftrightarrow \bigcirc\neg done_P) \wedge \bigcirc(msg_P = m)$. The abbreviations $done_C?$ and $done_C?m$ are defined analogously. Then, the desired requirement is specified by the formula

$$\varphi_{resp}: \bigwedge m \in \mathbb{M}.\, \Box\, [\, done_P?m \rightarrow \Diamond\, done_C?m\,].$$

Verify that the fair module *FairSyncMsg* satisfies the requirement φ_{resp}.

Let us now consider the copy-requirement that, in every round i, if $\overline{a}$ denotes the (finite) sequence of messages produced by the producer so far, then (1) the sequence of messages consumed by the consumer upto round i is a prefix of $\overline{a}$, and (2) there exists a later round j such that the sequence of messages consumed by the consumer upto round j equals $\overline{a}$. The former is a safety requirement, while the latter is a liveness requirement. The LTL formula φ_{resp} is only an approximation to the liveness part of the copy-requirement. It turns out that the copy-requirement is not expressible in LTL. We can approximate it by verifying φ_{resp}, and many additional weaker requirements such as

$$\bigwedge m \in \mathbb{M}.\, [\, \Box\Diamond\, done_P?m \leftrightarrow \Box\Diamond\, done_C?m\,],$$

which requires that the producer produces infinitely many messages with value m iff the consumer consumes infinitely many messages with value m, and

$$\bigwedge m \in \mathbb{M}.\, [\, (\neg done_C?m)\ \mathcal{W}\ done_P?m\,]$$

which requires that the consumer does not consume a message with value m unless at least one such message is produced by the producer.

A requirement stronger than the copy-requirement stipulates strict alternation between production and consumption starting with the production, and is expressed by the LTL formula $\varphi_{alternate}$:

$$\begin{array}{ll} & (done_C?\ \mathcal{W}\ done_P?) \\ \wedge & \wedge m \in \mathbb{M}.\, \Box[\, done_P?m \rightarrow \bigcirc((\neg done_P?)\,\mathcal{U}\, done_C?m)\,] \\ \wedge & \wedge m \in \mathbb{M}.\, \Box[\, done_C? \rightarrow \bigcirc((\neg done_C?)\,\mathcal{W}\, done_P?)\,] \end{array}$$

The fair module *FairSyncMsg* does not satisfy the requirement $\varphi_{alternate}$ even though *FairSyncMsg* satisfies the copy-requirement. This is because the producer may produce two messages before the consumer has consumed any message. ■

Exercise 12.1 {} [] Design a monitor module *CopyMonitor* whose variables keep track of the produced and consumed messages such that the safety aspect of the copy requirement reduces to an invariant verification problem for the compound module *SyncMsg* $\|$ *CopyMonitor* and the liveness aspect of the copy requirement reduces to a response verification problem for the compound fair module *FairSyncMsg* $\|$ *CopyMonitor*. ■

Like μ-calculus, fairness requirements can be specified within LTL. Let K be an observation structure, and consider a Büchi constraint specified by the observation predicate p. Then, an ω-trajectory $\underline{s}$ is fair iff the ω-trace $\langle\!\langle \underline{s} \rangle\!\rangle$ satisfies the LTL formula $\Box\Diamond p$. Suppose the fairness assumption requires fairness with respect to the action α specified as $[\![p \wedge q']\!]$ for two observation predicates p and q. Then, α-fair ω-trajectories are precisely those satisfying the LTL formula $\Box\Diamond(p \wedge \bigcirc q)$.

Now consider the Streett constraint (p, q) specified by two observation predicates p and q. Fairness of an ω-trajectory with respect to such a Streett constraint is specified by the LTL formula $\Box\Diamond p \rightarrow \Box\Diamond q$. Fairness with respect to multiple Streett constraints corresponds to conjunction of LTL formulas corresponding to individual Streett constraints.

Proposition 12.1 [Fairness specification in LTL] *Given a fair module $\mathcal{P}$, there exists an* LTL *formula $\varphi_{\mathcal{P}}$ such that an ω-trajectory $\underline{s}$ of the module $\mathcal{P}$ is a fair trajectory iff $\underline{s} \models \varphi_{\mathcal{P}}$.*

To verify all the fair trajectories of a fair module $\mathcal{P}$ satisfy a requirement ψ, we can verify whether all trajectories of the underlying module satisfy the implication $\varphi_{\mathcal{P}} \rightarrow \psi$.

Example 12.4 [Specifying fairness assumption in LTL] Recall the fairness constraints of the module *FairSyncMutex* from Figure 11.4. Requirement of weak fairness with respect to the update choices α_1 and α_2 is specified by the LTL formula $\varphi_{FairSyncMsg}$:

$$\Box\Diamond\, pc_1 \neq inC \;\wedge\; \Box\Diamond\, pc_2 \neq inC$$

Consider the specification φ_{sf} of starvation freedom from Example 12.2. The module *SyncMutex* does not satisfy the specification φ_{sf}, but satisfies the formula $\varphi_{FairSyncMutex} \rightarrow \varphi_{sf}$. ■

Exercise 12.2 {} [] Recall the fairness constraints of the module *FairPete* from Figure 11.5. Write down the LTL formula $\varphi_{FairPete}$ that captures the weak-fairness constraints of *FairPete*, and verify that the module *Pete* satisfies the LTL formula $\varphi_{FairPete} \rightarrow \varphi_{sf}$. ■

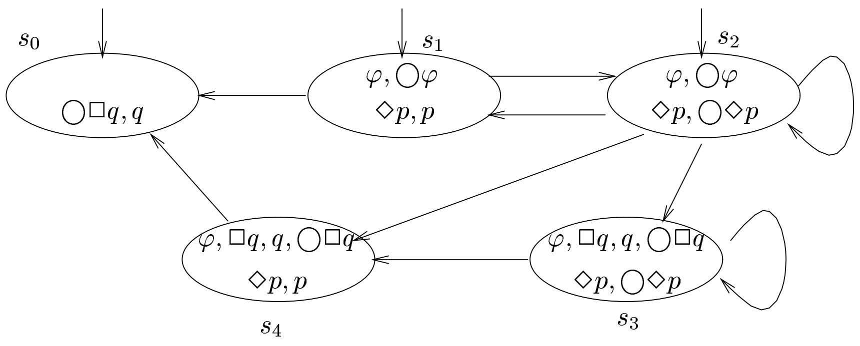

Figure 12.1: Tableau construction for $(\Diamond p)\,\mathcal{U}\,(\Box q)$

12.2 Decision procedure

In this section, we give an algorithm for constructing, given an LTL formula φ, a Büchi automaton accepting the set $\mathcal{L}_\varphi$ of ω-words satisfying φ. This construction leads to a model checking algorithm for LTL.

12.2.1 Tableau Decision Procedure

Let φ be an LTL formula. We wish to construct a Büchi automaton $\mathcal{M}_\varphi$ over the alphabet A_φ such that an ω-word $\underline{a}$ is accepted by $\mathcal{M}_\varphi$ iff $\underline{a} \models \varphi$. States of the desired automaton are sets of subformulas of φ. Such an automaton is called a *tableau*.

Sample construction

To illustrate the principles of the tableau construction, let us consider the LTL formula $\varphi = (\Diamond p)\,\mathcal{U}\,(\Box q)$. The states of the tableau are collections of LTL formulas derived from φ. Each state s is to a set of formulas, and we would like to ensure that every formula contained in the state s is satisfied by every source-s ω-trajectory in the tableau. The initial states of the tableau are required to contain the given formula φ. From the semantics of the until-connective, an initial state satisfies φ if either (1) it satisfies $\Box q$, or (2) it satisfies both $\bigcirc\varphi$ and $\Diamond p$. In the former case, to satisfy $\Box q$, the initial state should also satisfy q as well as $\bigcirc\Box q$, and this gives the initial state $s_0 = \{\varphi, \Box q, \bigcirc\Box q, q\}$ (see Figure 12.1). In the latter case, $\Diamond p$ can be satisfied by either p, or by $\bigcirc\Diamond p$. The corresponding initial states are $s_1 = \{\varphi, \bigcirc\varphi, \Diamond p, p\}$ and $s_2 = \{\varphi, \bigcirc\varphi, \Diamond p, \bigcirc\Diamond p\}$.

To obtain successors of a state, we examine the next-formulas in the state. For every $\bigcirc\psi$ contained in the current state, the successor should contain ψ. Since s_0 contains $\bigcirc\Box q$, its successor is required to contain $\Box q$, and hence, we add a transition from s_0 to itself. The successors of s_1 are required to contain φ.

Since φ can be satisfied in three ways, all the initial states are successors of s_1. The successors of s_2 are required to contain both φ and $\Diamond p$. The formula φ can be satisfied in three ways, while $\Diamond p$ can be satisfied either by p or by $\bigcirc\Diamond p$. Continuing in this manner, we get the tableau of Figure 12.1 with five states.

We would like to ensure that if $\underline{t}$ is an infinite trajectory in the tableau, then for every position $i \geq 0$ and every formula $\psi \in t_i$, $i \models_{\underline{t}} \psi$. This is not quite true yet. For instance, in the ω-trajectory s_2^ω that corresponds to looping forever at the state s_2, every state contains $\Diamond p$, but no state satisfies p. Intuitively, along this ω-trajectory the choice to satisfy $\Diamond p$ is postponed forever. This can be avoided by adding Büchi constraints. In this example, we need two Büchi constraints since there are two eventualities. The first constraint requires that to satisfy φ, one must satisfy $\Box q$ eventually. This is expressed by the Büchi constraint $\sigma_\varphi = \{s_0, s_3, s_4\}$ containing states that contain $\Box q$. The second constraint requires that to satisfy $\Diamond p$, one must satisfy p eventually. This is expressed by the Büchi constraint $\sigma_{\Diamond p} = \{s_0, s_1, s_4\}$ containing states that either contain p or do not contain $\Diamond p$. Verify that, for the tableau of Figure 12.1 together with the multi-Büchi assumption $\{\sigma_\varphi, \sigma_{\Diamond p}\}$, the set of fair trajectories corresponds to the set of ω-words that satisfy φ.

In summary, in a tableau construction, states are subsets of formulas. Each formula stipulates requirements concerning other formulas that must be satisfied in the current state. The transition relation ensures propagation of the next-formulas from one state to its successor. Fairness constraints ensure eventual fulfillment of eventuality- or until-formulas. We proceed to formalize the tableau construction. In our example, we treated formulas that do not appear in a state as "don't care." For instance, in the initial state s_0, there is no mention of the formulas $\Diamond p$ and p. In the formal construction, each state assigns a truth to every subformula.

Closure

The *closure* $Sub(\varphi)$ of the LTL formula φ is defined inductively as

$$\begin{array}{lcl}
Sub(p) & = & \{p\}; \\
Sub(\varphi_1 \vee \varphi_2) & = & \{\varphi_1 \vee \varphi_2\} \cup Sub(\varphi_1) \cup Sub(\varphi_2); \\
Sub(\neg\varphi) & = & \{\neg\varphi\} \cup Sub(\varphi); \\
Sub(\bigcirc\varphi) & = & \{\bigcirc\varphi\} \cup Sub(\varphi); \\
Sub(\varphi_1 \mathcal{U} \varphi_2) & = & \{\varphi_1 \mathcal{U} \varphi_2, \bigcirc(\varphi_1 \mathcal{U} \varphi_2)\} \cup Sub(\varphi_1) \cup Sub(\varphi_2).
\end{array}$$

Notice that the closure of a formula φ contains more than the syntactic subformulas of φ, namely, if an until-formula $\varphi_1 \mathcal{U} \varphi_2$ is in the closure, then so is the next-formula $\bigcirc(\varphi_1 \mathcal{U} \varphi_2)$.

Proposition 12.2 [Size of closure] *For every* LTL *formula* φ*,* $\varphi \in Sub(\varphi)$ *and* $|Sub(\varphi)| \leq 2|\varphi|$.

Tableau

A subset $s \subseteq Sub(\varphi)$ of the closure of φ is *consistent* if the following conditions are satisfied:

if $(\chi_1 \vee \chi_2) \in Sub(\varphi)$ then	$(\chi_1 \vee \chi_2) \in s$	iff	$\chi_1 \in s$ or $\chi_2 \in s$,
if $\neg\chi \in Sub(\varphi)$ then	$\neg\chi \in s$	iff	$\chi \notin s$,
if $(\chi_1 \mathcal{U} \chi_2) \in Sub(\varphi)$ then	$(\chi_1 \mathcal{U} \chi_2) \in s$	iff	either $\chi_2 \in s$ or both $\chi_1 \in s$ and $\bigcirc(\chi_1 \mathcal{U} \chi_2) \in s$.

LTL TABLEAU

Given an LTL formula φ the *φ-tableau* is the multi-Büchi automaton $\mathcal{M}_\varphi$:

- The state-space of $\mathcal{M}_\varphi$ is the set Σ_φ of consistent subsets of $Sub(\varphi)$.
- The transition relation of $\mathcal{M}_\varphi$ is the relation $\rightarrow_\varphi$: for $s, t \in \Sigma_\varphi$, $s \rightarrow_\varphi t$ if for all formulas $\bigcirc\chi \in Sub(\varphi)$, $\bigcirc\chi \in s$ iff $\chi \in t$.
- A state $s \in \Sigma_\varphi$ is initial iff $\varphi \in s$.
- The set of observations is the set A_φ of valuations of the atomic predicates in φ.
- The observation of a state s is the set of atomic predicates in s (that is, $\langle\langle s \rangle\rangle \models p$ iff $p \in s$).
- For each until-formula $(\chi_1 \mathcal{U} \chi_2) \in Sub(\varphi)$, the fairness assumption of $\mathcal{M}_\varphi$ contains the Büchi constraint

$$\{s \in \Sigma_\varphi \mid \chi_2 \in s \text{ or } (\chi_1 \mathcal{U} \chi_2) \notin s\}.$$

Proposition 12.3 [Correctness of tableau construction] *For every* LTL *formula φ, $\mathcal{L}_\varphi$ equals $\mathcal{L}_{\mathcal{M}_\varphi}$.*

Proof. Let φ be an LTL formula, and let $\underline{a}$ be an ω-word over the set A_φ of observations. Suppose $\underline{a} \models \varphi$. For $i \geq 0$, let $s_i \subseteq Sub(\varphi)$ be the set $\{\chi \in Sub(\varphi) \mid i \models_{\underline{a}} \chi\}$ of formulas true at position i in $\underline{a}$. From the definitions, it follows that (1) for all i, the set s_i is consistent, (2) for all i, $s_i \rightarrow_\varphi s_{i+1}$, (3) the set s_0 is an initial state of $\mathcal{M}_\varphi$, (4) for all i, the observation of the state s_i in $\mathcal{M}_\varphi$ is a_i, and (5) for each $(\chi_1 \mathcal{U} \chi_2) \in Sub(\varphi)$, if $i \models_{\underline{a}} \chi_1 \mathcal{U} \chi_2$ for infinitely many positions i, then $j \models_{\underline{a}} \chi_2$ for infinitely many positions j. It follows that $\underline{s}$ is a fair initialized ω-trajectory of the tableau $\mathcal{M}_\varphi$, and $\underline{a}$ belongs to $\mathcal{L}_{\mathcal{M}_\varphi}$.

Now consider an initialized fair ω-trajectory $\underline{s}$ of $\mathcal{M}_\varphi$. Let $\underline{a}$ be the corresponding ω-trace. We want to establish that for all $\chi \in Sub(\varphi)$, for all $i \geq 0$, $\chi \in s_i$ iff $i \models_{\underline{a}} \chi$. The proof is by induction on the structure of χ, and is left as an exercise. It follows that $\underline{a} \models \varphi$. ∎

The number of states of the tableau of an LTL formula is exponential in the length of the formula: for an LTL formula φ of length k, the automaton $\mathcal{M}_\varphi$ has at most 4^k states and k Büchi constraints. Checking satisfiability of the LTL formula φ corresponds to checking whether the fair language of the automaton $\mathcal{M}_\varphi$ is nonempty, and thus, can be solved in time $\mathbf{2}^{O(|\varphi|)}$. Checking validity of the LTL formula corresponds to checking satisfiability of the negated formula $\neg\varphi$, and thus, can also be solved in time $\mathbf{2}^{O(|\varphi|)}$.

Exercise 12.3 {} [] In our definition of the tableau $\mathcal{M}_\varphi$, every state assigns a truth value to every formula in $Sub(\varphi)$. As indicated in our example (see Figure 12.1), not every formula needs to be evaluated in every state. Develop an algorithm to construct a tableau for the input formula that considers the formulas in the closure only as needed. ■

12.2.2 LTL Model Checking

Let $\mathcal{K}$ be a fair structure, and let φ be an LTL formula. The model checking problem $(\mathcal{K}, \varphi)$ corresponds to verifying that the ω-language $\mathcal{L}_\mathcal{K}$ is contained in the ω-language $\mathcal{L}_\varphi$, which, by the tableau-construction, corresponds to the ω-language inclusion problem $(\mathcal{K}, \mathcal{M}_\varphi)$. Observe that the tableau $\mathcal{M}_\varphi$ is a nondeterministic ω-automaton, and hence, solving the ω-language inclusion problem $(\mathcal{K}, \mathcal{M}_\varphi)$ is computationally hard, namely, exponential in the size of the structure $\mathcal{M}_\varphi$.xs However, we can avoid the complementation construction for ω-automata if, instead of constructing the tableau for φ, we construct the tableau $\mathcal{M}_{\neg\varphi}$ for the negated formula $\neg\varphi$. The ω-automaton $\mathcal{M}_{\neg\varphi}$ accepts all ω-words that do not satisfy the specification φ. Now, the fair structure $\mathcal{K}$ satisfies φ iff the intersection of the ω-languages of $\mathcal{K}$ and $\mathcal{M}_{\neg\varphi}$ is empty. This approach of negating the formula *before* applying the tableau-construction avoids the need for complementing the tableau.

Proposition 12.4 [LTL model checking] *The answer to the* LTL *model checking problem* $(\mathcal{K}, \varphi)$ *is* YES *iff the answer to the fair-emptiness problem* $\mathcal{K} \times \mathcal{M}_{\neg\varphi}$ *is* YES.

Since we already know how to obtain product of two ω-automata, and we know how to solve the fair emptiness problem for finite structures, we have an algorithm for LTL model checking.

Theorem 12.1 [LTL model checking] *If* φ *is an* LTL *formula of length* k *and* $\mathcal{K}$ *is a fair structure with* n *states,* m *transitions, and* ℓ *fairness constraints, then the* LTL *model checking problem* $(\mathcal{K}, \varphi)$ *can be solved in time* $O((m+n) \cdot \ell^2 \cdot k \cdot \mathbf{2}^{O(k)})$.

Remark 12.4 [LTL model checking of weak-fair structures] If $\mathcal{K}$ is a weak-fair structure, then so is the product $\mathcal{K} \times \mathcal{M}_{\neg\varphi}$. If $\mathcal{K}$ has ℓ weak-fairness constraints, and $Sub(\varphi)$ has k until-formulas, then the product has $\ell + k$ weak-fairness constraints. It follows that the the LTL model checking problem $(\mathcal{K}, \varphi)$, for a weak-fair structure, can be solved in time $O((m+n) \cdot (\ell + k) \cdot \mathbf{2}^{O(k)})$. ■

A variety of heuristics can be used to improve the computational requirements of LTL model checking. In particular, an on-the-fly representation of the structure is used, and the product with the tableau is generated only during the search.

Remark 12.5 [] Recall that the CTL model-checking problem $(\mathcal{K}, \phi)$, for a CTL formula ϕ of length k, can be solved in time $O((m+n) \cdot \ell^2 \cdot k)$. Thus, while CTL model checking is linear in the size of the formula, LTL model checking is exponential in the size of the formula. However, both model checking problems have identical model complexity. ■

If $\mathcal{P}$ is a propositional fair module with n boolean variables and φ is an LTL formula of length k, then the LTL verification problem $(\mathcal{P}, \varphi)$ can be solved in time exponential in $n + k$.

12.2.3 Complexity

The LTL satisfiability problem is to determine whether a given LTL formula φ is satisfiable. Checking satisfiability of φ corresponds to checking emptiness of the tableau $\mathcal{M}_\varphi$. Search for a reachable fair cycle in the tableau can be performed using space logarithmic in the number of states of the tableau, or equivalently, linear in the size of the formula φ. It follows that the LTL satisfiability problem is in PSPACE. It turns out that checking satisfiability of LTL formulas is PSPACE-hard.

Theorem 12.2 [LTL complexity] *The satisfiability and the validity problems for* LTL *are PSPACE-complete.*

Proof. An LTL formula φ is satisfiable iff the formula $\neg\varphi$ is valid. It remains to be shown that the satisfiability problem is PSPACE-hard. Proof to be added. ■

Hardness of the satisfiability problem implies a lower bound for the model checking problem also.

Theorem 12.3 [LTL model checking comlexity] *The* LTL *model checking problem $(\mathcal{K}, \varphi)$, for a finite fair structure $\mathcal{K}$, is PSPACE-complete.*

Proof. The LTL model checking problem $(\mathcal{K}, \varphi)$, for a finite fair structure $\mathcal{K}$, reduces to searching for a reachable fair cycle of the product $\mathcal{K} \times \mathcal{M}_{\neg\varphi}$. The search can be performed in space logarithmic in the number of states of the product, and PSPACE upper bound follows.

For lower bound, we reduce the LTL validity problem to LTL model checking. ■

Remark 12.6 [] The model complexity of the LTL model checking problem is NLOGSPACE. The complexity of the LTL verification problem $(\mathcal{P}, \varphi)$, for a propositional fair module $\mathcal{P}$, is PSPACE. ■

Exercise 12.4 {} [] Show that the decision and model checking problems for LTL without the next operator are still PSPACE-hard. ■

Exercise 12.5 {} [] Let $\text{LTL}^{\diamond}$ result from LTL by replacing the until operator $\mathcal{U}$ with the eventually operator $\diamond$. Prove that the decision problem for $\text{LTL}^{\diamond}$ is still PSPACE-hard. ■

Exercise 12.6 {} [] Show that if an $\text{LTL}^{\diamond}$ formula φ without next operators is satisfiable, then it is satisfiable by an ω-word of the form $\overline{a}_1\overline{a}_2^{\omega}$ such that $|\overline{a}_1\overline{a}_2| \leq |\varphi|$. Then prove that the decision problem for $\text{LTL}^{\diamond}$ without the next operator is NP-complete. ■

Exercise 12.7 {} [LTL with past operators] The syntax of *Past* LTL is defined as

$$\varphi \ ::= \ p \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1\mathcal{U}\varphi_2 \mid \ominus\varphi \mid \varphi_1\mathcal{S}\varphi_2.$$

The semantics of the *previous* and *since* operators are defined as

$i \models_{\overline{a}} \ominus\varphi$ iff $i > 0$ and $i-1 \models_{\overline{a}} \varphi$;
$i \models \varphi_1\mathcal{S}\varphi_2$ iff for some $j \leq i$, $j \models_{\overline{a}} \varphi_2$ and for all $j < k \leq i$, $k \models_{\overline{a}} \varphi_1$.

Give a tableau-based decision procedure for Past LTL and then prove that the decision problem for Past LTL is complete for PSPACE. ■

Exercise 12.8 {} [] The syntax of *Counting* LTL is defined as

$$\varphi \ ::= \ p \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc^n\varphi \mid \varphi_1\mathcal{U}\varphi_2$$

for natural numbers n (represented in logarithmic notation, e.g., binary or decimal). The semantics of the *counting* operator is defined as

$i \models_{\overline{a}} \bigcirc^n\varphi$ iff $i+n \models_{\overline{a}} \varphi$.

Give a tableau-based decision procedure for Counting LTL and then prove that the decision problem for Counting LTL is complete for EXPSPACE. ■

12.3 Expressiveness

12.3.1 Linear-time versus branching-time

How do the expressive powers of the branching-time logic CTL and the linear time logic LTL compare? We have already seen that CTL cannot express operators such as $\Box\Diamond$, and thus, cannot be more expressive than LTL. For the converse, while the state equivalence induced by CTL coincides with bisimilarity, the state equivalence induced by LTL coincides with trace-equivalence:

Proposition 12.5 [State equivalence of LTL] *Trace-equivalence is a fully abstract semantics for* LTL *over observation structures.*

Since CTL can distinguish between two states that are trace-equivalent, but not bisimilar, LTL cannot be more expressive that CTL.

Proposition 12.6 [Expressive power of LTL vs. CTL] *The expressive powers of the temporal logics* CTL *and* LTL *are incomparable. In particular, no* LTL *formula is equivalent to the* CTL *formula* $\forall\bigcirc\exists\bigcirc p$, *and no* CTL *formula is equivalent to the* LTL *formula* $\Box\Diamond p$.

The deficiency of LTL compared to CTL is the lack of existential quantification over trajectories, while the deficiency of CTL compared to LTL is the inability to nest temporal connectives to express requirements regarding a fixed trajectory. This motivates the definition of a temporal logic that is more expressive than both CTL and LTL.

Before we define the combination of CTL and LTL, let us understand the distinction between state formulas and trajectory formulas. While the formulas of a (fair) state logic—*state formulas*—are interpreted over the states of a (fair) observation structure, the formulas of a trajectory logic—*trajectory formulas*—are interpreted over the ω-trajectories of a (fair) observation structure. We can view CTL as a two-sorted logic with state formulas ϕ and trajectory formulas φ:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\varphi$$

$$\varphi ::= \bigcirc\phi \mid \phi_1\mathcal{U}\phi_2 \mid \Box\phi$$

where p is an atomic formula. On the other hand, we can view LTL as a two-sorted logic with state formulas ϕ and trajectory formulas φ:

$$\phi ::= \forall\varphi$$

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \phi_1\mathcal{U}\phi_2$$

where p is an atomic formula. The logic CTL* allows the state-formulas as in CTL and trajectory formulas as in LTL.

TEMPORAL LOGIC CTL*: SYNTAX

The formulas of CTL* are defined inductively by the two-sorted grammar with state formulas ϕ and trajectory formulas φ:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\varphi$$

$$\varphi ::= \phi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

where p is an atomic formula.

Given a fair structure $\mathcal{K}$ whose observations are valuations to the atomic predicates, state-formulas of CTL* are interpreted at states of $\mathcal{K}$, while the trajectory-formulas of CTL* are interpreted at positions of the fair trajectories of $\mathcal{K}$.

TEMPORAL LOGIC CTL*: SEMANTICS

Let $\mathcal{K} = (K, F)$ be a fair structure. For each state s of $\mathcal{K}$,

$$\begin{array}{lll}
s \models_{\mathcal{K}} p & \text{iff} & \langle\!\langle s \rangle\!\rangle \models p; \\
s \models_{\mathcal{K}} \neg\phi & \text{iff} & s \not\models_{\mathcal{K}} \phi; \\
s \models_{\mathcal{K}} \phi_1 \vee \phi_2 & \text{iff} & s \models_{\mathcal{K}} \phi_1 \text{ or } s \models_{\mathcal{K}} \phi_2; \\
s \models_{\mathcal{K}} \exists\varphi & \text{iff} & \text{there is a source-}s\ F\text{-fair } \omega\text{-trajectory } \underline{s} \\
 & & \text{of } K \text{ such that } (\underline{s}, 0) \models_{\mathcal{K}} \varphi.
\end{array}$$

For each ω-trajectory $\underline{s}$ of $\mathcal{K}$ and each position $i \geq 0$,

$$\begin{array}{lll}
(\underline{s}, i) \models_{\mathcal{K}} \phi & \text{iff} & s_i \models_{\mathcal{K}} \phi; \\
(\underline{s}, i) \models_{\mathcal{K}} \neg\varphi & \text{iff} & (\underline{s}, i) \not\models_{\mathcal{K}} \varphi; \\
(\underline{s}, i) \models_{\mathcal{K}} \varphi_1 \vee \varphi_2 & \text{iff} & (\underline{s}, i) \models_{\mathcal{K}} \varphi_1 \text{ or } (\underline{s}, i) \models_{\mathcal{K}} \varphi_2; \\
(\underline{s}, i) \models_{\mathcal{K}} \bigcirc\varphi & \text{iff} & (\underline{s}, i+1) \models_{\mathcal{K}} \varphi; \\
(\underline{s}, i) \models_{\mathcal{K}} \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \text{there exist } j \geq i \text{ such that } (\underline{s}, j) \models_{\mathcal{K}} \varphi_2 \\
 & & \text{and for all } i \leq k < j,\ (\underline{s}, k) \models_{\mathcal{K}} \varphi_1.
\end{array}$$

The fair structure $\mathcal{K}$ satisfies the CTL* formula ϕ if $s \models_{\mathcal{K}} \phi$ for every initial state s of $\mathcal{K}$.

The following operators are defined in CTL*:

$$\begin{array}{lll}
\forall\varphi & \text{for} & \neg\exists\neg\phi; \\
\Diamond\varphi & \text{for} & \mathit{true}\mathcal{U}\varphi; \\
\Box\varphi & \text{for} & \neg\Diamond\neg\varphi; \\
\varphi_1 \mathcal{W} \varphi_2 & \text{for} & \varphi_1 \mathcal{U} \varphi_2 \vee \Box\varphi_1.
\end{array}$$

Remark 12.7 [CTL* vs. LTL and CTL] Since every CTL formula is a CTL* formula, and for every LTL formula φ, the equivalent CTL* formula is $\forall\varphi$, it follows that CTL* is more expressive power than both LTL and CTL. ■

While the expressive power of CTL* is more than CTL, the equivalence induced by CTL* coincides with bisimilarity, and thus, its distinguishing power coincides with CTL.

Proposition 12.7 [Equivalence induced by CTL*] *Bisimilarity is a fully abstract semantics of* CTL* *over observation structures.*

Exercise 12.9 {} [] Prove Proposition 12.7. ■

Exercise 12.10 {} [] The formulas of CTL^+ are defined inductively by the two-sorted grammar

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\varphi$$

$$\varphi ::= \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\phi \mid \phi_1 \mathcal{U} \phi_2$$

where p is an atomic formula. For example, for two propositions a and b, the CTL^+ formula $\exists(\Diamond a \wedge \Diamond b)$ is equivalent to the CTL formula

$$\exists\Diamond(a \wedge \exists\Diamond b) \vee \exists\Diamond(b \wedge \exists\Diamond a).$$

Give a systematic construction that yields for each CTL^+ formula ϕ an equivalent CTL formula ϕ^-. For your construction, what is the blowup in the size of the formula? (If ϕ has length k, give an asymptotic bound for the length of ϕ^- as a function of k.) ■

Exercise 12.11 {} [] Let $\mathrm{CTL}_\Diamond$ be obtained from CTL by replacing the binary operator $\exists\mathcal{U}$ with the unary operator $\exists\Diamond$, and let $\mathrm{CTL}^+_\Diamond$ be obtained from CTL^+ by replacing the binary operator $\mathcal{U}$ with the unary operator $\Diamond$. Consider two propositions a and b. Prove that no $\mathrm{CTL}_\Diamond$ formula is equivalent to the $\mathrm{CTL}^+_\Diamond$ formula $\exists(\Box a \wedge \Diamond b)$ over finite observation structures, and that no $\mathrm{CTL}^+_\Diamond$ formula is equivalent to the CTL formula $a\exists\mathcal{U}b$. We conclude that $\mathrm{CTL}_\Diamond \prec \mathrm{CTL}^+_\Diamond \prec$ CTL $\equiv \mathrm{CTL}^+ \prec \mathrm{CTL}^*$. ■

CTL* model checking

Theorem 12.4 [CTL* model checking] *If ϕ is a* CTL* *formula of length k and $\mathcal{K}$ is a fair structure with n states, m transitions, and ℓ fairness constraints, then the* CTL* *model checking problem $(\mathcal{K}, \phi)$ can be solved in time $O((m+n) \cdot \ell^2 \cdot \mathbf{2}^{O(k)})$.*

Remark 12.8 [Space complexity of CTL*] For finite fair structures, the CTL* model checking problem is PSPACE-complete. The structure complexity of CTL* model checking is NLOGSPACE. Thus, the space complexity of CTL* model checking coincides with the space complexity of LTL model checking. The CTL* verification problem $(\mathcal{P}, \phi)$ is PSPACE-complete. ■

12.3.2 LTL versus ω-automata

The tableau construction establishes that, for every LTL formula φ, the ω-language $\mathcal{L}_\varphi$ is ω-regular. The converse does not hold. In particular, the property that requires p to be true at every even position is not specifiable in LTL, while it is speciable using automata as indicated in Figure 9.6.

Proposition 12.8 [LTL cannot express even] *For $A = \{a, b\}$, the ω-regular language $\mathcal{L}_{even}$ that contains the ω-word $\underline{c}$ iff $c_i = a$ for all even numabers i, is not expressible in* LTL.

Corollary 12.1 [LTL VS. LAL] *The live automaton logic* LAL *is more expressive than* LTL.

While LAL is more expressive than LTL, the distinguishing powers of the two logics coincide, both logics induce trace-equivalence over observation structures.

Exercise 12.12 {} [] The syntax of *Periodic* LTL is defined as

$$\varphi \ ::= \ p \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U}_n \varphi_2$$

for natural numbers n. The semantics of the *periodic-until* operator is defined as

$$i \models_{\overline{a}} \varphi_1 \mathcal{U}_n \varphi_2 \text{ iff for some } j \geq 0,\ i + jn \models_{\overline{a}} \varphi_2 \text{ and for all } 0 \leq k < j,\ i + kn \models_{\overline{a}} \varphi_1.$$

(1) Express the property that "proposition x is true at every even position of an observation sequence" in Periodic LTL. (2) Give a tableau-based decision procedure for Periodic LTL. (3) What is the complexity of the decision problem for Periodic LTL if numerals are represented in unary (binary, respectively)? ■

Exercise 12.13 {} [Linear-time μ-calculus] The formulas of the *Linear μ-calculus* LTμ are defined as

$$\varphi \ ::= \ p \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \mu X.\, \varphi' \mid X$$

where p is an observation predicate, $X \in \mathcal{P}$ is a formula variable, and each free occurrence of X in φ' occurs within an even number of negations. The linear μ-calculus is interpreted over infinite observation sequences. Give a formal definition of the semantics of LTμ such that (1) the LTL formula $\varphi_1 \mathcal{U} \varphi_2$ is equivalent to the LTμ formula $(\mu X.\, \varphi_2 \vee (\varphi_1 \wedge \bigcirc X))$ and (2) the LTμ formula $(\nu X.\, y \wedge \bigcirc \bigcirc X)$ is satisfied by the ω-word $\underline{a}$ iff the proposition y is true at every even-numbered position of $\overline{a}$. When defining the semantics of LTμ, you need to show that all required fixpoints exist (What is the underlying c.p.o.? Why are all definable functions monotonic?) and that (1) and (2) are indeed the case. Then prove that all definable functions are continuous. ■

Monadic second-order logic S1S

The fragment S1S of second-order logic is a classical notation to define ω-languages. The logic S1S allows first-order variables that range over nonnegative integers, and its terms are built from the first-order variables using the successor function "+1" that corresponds to adding 1. The formulas are built using logical connectives, first-order and second-order quantifiers, comparing terms using the ordering $<$, and second-order unary variables.

Formally, the set of terms of S1S is generated by the grammar

$$e \;:=\; 0 \mid i \mid e+1,$$

where i is a first-order variable. The set of formulas of S1S is generated by the grammar

$$\varphi \;:=\; p(e) \mid e < e \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists i.\, \varphi \mid \exists p.\, \varphi,$$

where p is a second-order variable.

Formulas of S1S are evaluated with respect to environments that map first-order variables to nonnegative integers and second-order variables to sets of nonnegative integers. Let $\mathcal{E}$ be an environment that maps first-order variables to $\mathbb{N}$ and second-order variables to $\mathbf{2}^{\mathbb{N}}$. Then, $\mathcal{E}$ maps terms of S1S to $\mathbb{N}$: $\mathcal{E}(0) = 0$ and $\mathcal{E}(e+1) = \mathcal{E}(e)+1$. The satisfaction relation $\models_{\mathcal{E}}$ for the formulas of S1S is defined inductively:

$$\begin{array}{lll}
\models_{\mathcal{E}} p(e) & \text{iff} & \mathcal{E}(e) \in \mathcal{E}(p); \\
\models_{\mathcal{E}} e_1 < e_2 & \text{iff} & \mathcal{E}(e_1) < \mathcal{E}(e_2); \\
\models_{\mathcal{E}} \neg\varphi & \text{iff} & \not\models_{\mathcal{E}} \varphi; \\
\models_{\mathcal{E}} \varphi_1 \vee \varphi_2 & \text{iff} & \models_{\mathcal{E}} \varphi_1 \text{ or } \models_{\mathcal{E}} \varphi_2; \\
\models_{\mathcal{E}} \exists i.\, \varphi & \text{iff} & \text{for some } j \in \mathbb{N},\ \models_{\mathcal{E}[i:=j]} \varphi; \\
\models_{\mathcal{E}} \exists p.\, \varphi & \text{iff} & \text{for some } \sigma \subseteq \mathbb{N},\ \models_{\mathcal{E}[p:=\sigma]} \varphi.
\end{array}$$

The unary predicates in S1S formulas can be viewed as boolean variables. If X consists of boolean variables, and $\underline{a}$ is an ω-word over the valuations for X, then, for every $x \in X$, the ω-word $\underline{a}$ specifies the set $x[\underline{a}] = \{i \geq 0 \mid a_i \models x\}$ of positions. Consequently, a formula φ of S1S can be evaluated with respect to ω-words over observations that evaluate the second-order variables in φ. For instance, the S1S formula

$$\forall i.\, (p(i) \;\rightarrow\; \exists j.\, (i \leq j \;\wedge\; q(j)))$$

specifies the ω-language corresponding to the LTL formula

$$\Box(p \;\rightarrow\; \Diamond q).$$

Let φ be an S1S formula whose free second-order variables are in X, and let $\underline{a}$ be an ω-word over the alphabet Σ_X. Then, $\underline{a} \models \varphi$ iff $\models_{\mathcal{E}} \varphi$ for every environment $\mathcal{E}$ such that for all $p \in X$ and all $i \in \mathbb{N}$, $i \in \mathcal{E}(p)$ iff $a_i \models p$. The ω-language $\mathcal{L}_\varphi$ consists of ω-words $\underline{a}$ over Σ_X such that $\underline{a} \models \varphi$.

Exercise 12.14 {} [] Define a mapping from LTL formulas to S1S formulas that preserves the ω-language. ■

An outstanding theorem due to Büchi establishes that the expressive power of S1S and Büchi automata coincide:

Theorem 12.5 [Büchi Theorem on S1S vs. automata] *Let X be a finite set of boolean variables, and let $\mathcal{L}$ be an ω-language over the alphabet Σ_X. Then, $\mathcal{L}$ is ω-regular iff there exists an* S1S *formula φ such that $\mathcal{L}_\varphi = \mathcal{L}$.*

Remark 12.9 [] The satisfiability and the validity problems for S1S are of nonelementary complexity. ■

First-order fragment of S1S

The logic $\mathrm{S1S}^{fo}$ consists of the fragment of S1S that disallows quantification over second-order variables. Formally, the set of formulas of $\mathrm{S1S}^{fo}$ is generated by the grammar

$$\varphi \ := \ p(e) \mid e < e \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists i.\, \varphi,$$

where i is a first-order variable, p is a second-order variable and e is a term of S1S. It turns out that the first-order fragment of S1S precisely captures the expressiveness of LTL.

Theorem 12.6 [Expressiveness of LTL] *For an ω-language $\mathcal{L}$, there exists an* LTL *formula φ with $\mathcal{L}_\varphi = \mathcal{L}$ iff there exists a formula ψ of* $\mathrm{S1S}^{fo}$ *with $\mathcal{L}_\psi = \mathcal{L}$.*