Basics of Model Checking

Sylvain Schmitz
LSV, ENS Cachan & CNRS & INRIA
September 21, 2015 (r6396M)

These rough notes cover the contents of the first half of an introductory course on verification, also known as **MPRI 1-22**: *Basics of Verification*. These notes are compiled from various sources; most notably:

Baier, C. and Katoen, J.P., 2008. *Principles of Model Checking*. MIT Press.

Diekert, V. and Gastin, P., 2008. First-order definable languages. In Flum, J., Grädel, E., and Wilke, T., editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press. http://dare.uva.nl/document/154959#page=262.

Schnoebelen, Ph., 2003. The complexity of temporal logic model checking. In Balbiani, Ph., Suzuki, N.Y., Wolter, F., and Zakharyaschev, M., editors, *AiML'02), 4th Workshop on Advances in Modal Logics*, pages 393–436. King's College Publication. http://www.lsv.fr/Publis/PAPERS/PDF/Sch-aiml02.pdf. Cited on page 9.

Sistla, A.P. and Clarke, E.M., 1985. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749. doi:10.1145/3828.3837.

Cited on page 10.

Contents

Systems

Definition 0.1 (Transition System). A **transition system** is a tuple $\mathfrak{S} = (S, T, I, \mathrm{AP}, \ell)$ where

- $S = \{s_1, s_2, \dots\}$ is a set of states (finite or infinite)
- $T \subseteq S \times S$ is a set of transitions
- $I \subseteq S$ is a set of initial states
- $\mathrm{AP} = \{p, q, \dots\}$ is a countable set of atomic propositions
- $\ell : S \to 2^{\mathrm{AP}}$ is a labelling function.

A transition system is **finitely branching** if for every $s \in S$, the set $\{s' \in S \mid (s, s') \in T\}$ is finite. A transition system is **finite** if S is finite. A state s of a transition system is a **dead-end** if there are no a, s' such that $(s, a, s') \in T$. Most often, we will only consider transitions systems without dead-ends.

When specifying systems, we usually ignore the underlying set S of states, and only talk about the 'observable' part of the system, namely atomic propositions. There are several properties that one might want to verify on a given system. The first ones are *safety* properties, like 'no segmentation fault,' 'no division by zero,' etc. The question they ask is wether a set of states $\mathsf{BAD} \subseteq S$ intersect the set of states that are reachable from the initial states.

The lecture will focus on *temporal* properties, that differ from safety properties in the sense that they usually ask questions that do not necessarily reduce to reachability problems.

We distinguish two families of temporal properties:

- *linear-time* properties, that talk about the *infinite runs* (or *traces*) of the system, to be defined soon;
- *branching-time* properties, that talk about the *computation tree* of the system.

Definition 0.2 (Run). A **run** is a finite or infinite sequence $\sigma = s_0 s_1 s_2 \cdots$ such that $(s_i, s_{i+1}) \in T$ for all $i \geq 0$ and $s_0 \in I$.

Chapter 1

Logical Specifications

In the model-checking paradigm, the desired properties of a transition system must be expressed formally through logical formulæ. The logics we consider to this end do not talk directly about the system, e.g. do not express 'there is a state q and it has two transitions to q_1 and q_2 respectively,' which we take to be low-level implementation details. Instead, the formulæ allow to reason about the system's *behaviours*, like 'we can never see both propositions crit_1 and crit_2 hold simultaneously' or 'every request is eventually granted.' Such behaviours are defined formally using *time flows* associated to the system, see Section 1.1. We will then consider two logics on such time flows, namely first-order logic (Section 1.2) and modal logic (Section 1.3); the latter will be the basis for the more expressive *temporal logics* of **??**.

1.1 System Behaviours

We first define a general notion of time flows and temporal models. We shall see then two natural ways of associating time flows to transition systems as a means of representing their behaviours: *linear time* and *branching time*.

1.1.1 Time Flows

A **time flow**, also called a **temporal frame**, is a pair $\mathfrak{F} = (\mathbb{T}, <)$ where $\mathbb{T}$ is a nonempty set of *(time) points* and $<$ is an irreflexive transitive relation over $\mathbb{T}$.

Example 1.1 (Time Flows)**.** Two time flows are of particular interest in these notes for verification purposes:

- $(\mathbb{N}, <)$ can be used to represent some infinite run of a sequential system;
- infinite trees for run-trees of sequential systems, which allow to reason simultaneously about several possible behaviours of the system.

Some other time flows of interest include

- $(\{0, \ldots, n-1\}, <)$ for finite runs of sequential systems;
- $(\mathbb{R}, <)$ for runs of real-time sequential systems;
- partial orders (or Mazurkiewicz traces) for runs of distributed systems.

We might also want to reason about $(\mathbb{Z}, <)$, $(\mathbb{Q}, <)$, $(\omega^2, <)$, ...

We are actually interested in time flows where the points are decorated with **atomic propositions** from some countable set AP. We define accordingly a **temporal model** as a triple $\mathfrak{M} = (\mathbb{T}, <, h)$ where $(\mathbb{T}, <)$ is a time flow and $h\colon \mathrm{AP} \to 2^{\mathbb{T}}$ is an assignment of the atomic propositions. Thus, if p is an atomic proposition in AP, then $h(p) \subseteq \mathbb{T}$ provides the set of points in the time flow where p holds.

1.1.2 Linear Time Behaviours

Consider a transition system $\mathfrak{S} = (S, T, I, \mathrm{AP}, \ell)$. An infinite run $\sigma = s_0 s_1 \cdots$ with $(s_i, s_{i+1}) \in T$ for all i defines an **infinite word** $\ell(\sigma) \overset{\text{def}}{=} \ell(s_0)\ell(s_1)\cdots$ over $\Sigma \overset{\text{def}}{=} 2^{\mathrm{AP}}$, which can be seen as a **linear temporal model** $(\mathbb{N}, <, h)$ where $h(p) \overset{\text{def}}{=} \{i \in \mathbb{N} \mid p \in \ell(s_i)\}$.

We denote by

$$\mathrm{Runs}(\mathfrak{S}) \overset{\text{def}}{=} \{\ell(\sigma) \mid \sigma(0) \in I\} \subseteq \Sigma^\omega \tag{1.1}$$

the set of linear temporal models of $\mathfrak{S}$ that start in some initial state, i.e. some state in I.

Example 1.2. Consider the following transition system with $\mathrm{AP} \overset{\text{def}}{=} \{a, r, h\}$ standing respectively for 'acknowledgement,' 'request,' and 'halt.'

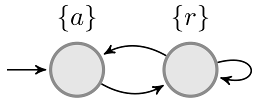

Figure 1.1: A finite transition system.

Its set of linear time behaviours is the language of infinite words $\big(\{a\}\{r\}^*\big)^\omega + \big(\{a\}\{r\}^*\big)^*\{r\}^\omega$, e.g. it contains among others the infinite words

$$\{a\}\{r\}\{a\}\{r\}\{a\}\{r\}\cdots = (\{a\}\{r\})^\omega\ , \tag{1.2}$$

$$\{a\}\{r\}\{a\}\{r\}\{r\}\{a\}\{r\}\{r\}\{r\}\cdots\{a\}\{r\}^i\cdots = \bigodot_{i>0}\{a\}\{r\}^i\ , \tag{1.3}$$

$$\{a\}\{r\}\{r\}\{r\}\{r\}\{r\}\cdots = \{a\}\{r\}^\omega\ . \tag{1.4}$$

Thus, although $\mathrm{Runs}(\mathfrak{S})$ is a regular language of infinite words (in a sense we will see in **??**), it contains some irregular words like (1.3).

1.1.3 Branching Time Behaviours

Consider again a transition system $\mathfrak{S} = (S, T, I, \mathrm{AP}, \ell)$. The **unfolding** of $\mathfrak{S}$ from a state $s \in S$ is the infinite transition system $\mathrm{tree}(s) \overset{\text{def}}{=} (S', T', \{s\}, \mathrm{AP}, \ell')$ with

$$S' \overset{\text{def}}{=} \{s_0 s_1 \cdots s_n \mid n \geq 0 \wedge s_0 = s \wedge \forall 0 \leq i < n\,.\,(s_i, s_{i+1}) \in T\}$$

the set of finite runs of $\mathfrak{S}$ from s,

$$\begin{aligned} T' &\overset{\text{def}}{=} \{(s_0 \cdots s_n, s_0 \cdots s_n s_{n+1}) \mid (s_n, s_{n+1}) \in T\}\ , \\ \ell'(s_0 \cdots s_n) &\overset{\text{def}}{=} \ell(s_n)\ . \end{aligned}$$

Put differently, $\mathrm{tree}(s)$ is an infinite unordered tree labelled by subsets of AP. It can be seen as a **branching temporal model** $(S', <, h)$ where $<$ is the transitive closure of T' and $h(p) \overset{\text{def}}{=} \{s_0 \cdots s_n \in S' \mid p \in \ell(s_n)\}$.

We let similarly

$$\mathrm{Trees}(\mathfrak{S}) \overset{\text{def}}{=} \{\mathrm{tree}(s) \mid s \in I\} \tag{1.5}$$

be the set of unfoldings of $\mathfrak{S}$ from its initial states.

Example 1.3. Consider again the transition system of Figure 1.1 in Example 1.2. It has a single initial state, hence its set of trees contains a single tree, depicted in Figure 1.2.

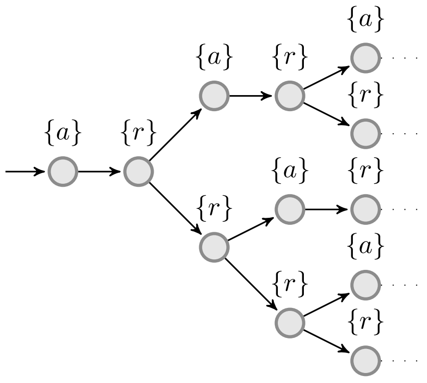

Figure 1.2: The run-tree of the transition system of Figure 1.1.

Two remarks are in order:

1. Each tree $\mathrm{tree}(s)$ for a state s is a *regular* infinite tree, meaning that it contains only finitely many non-isomorphic subtrees. In fact, $\mathrm{tree}(s)$ connects its root s with isomorphic copies of $\mathrm{tree}(s')$ for each s' in $T(s) \overset{\text{def}}{=} \{s' \in S \mid (s,s') \in T\}$. This should again be contrasted with the situation with $\mathrm{Runs}(\mathfrak{S})$, where individual runs like (1.3) do not necessarily have this property.

2. The set of runs of $\mathfrak{S}$ can still be extracted from $\mathrm{Trees}(\mathrm{S})$: formally, $\mathrm{Runs}(\mathrm{S}) = \bigcup_{s\in I} \mathrm{Runs}(\mathrm{tree}(s))$. The converse is in general not possible (van Glabbeek, 2001). In this sense, the set of trees is richer than the set of runs.

Remark 1.4 (Paths)**.** Consider the very specific case of a transition system with $|T(s)| = 1$ for all $s \in S$. Observe that, in that case, $\mathrm{tree}(s)$ is a linear temporal model, i.e. linear time and branching time coincide. If $\mathfrak{S}$ is finite, then, by the Pigeonhole Principle, $\mathrm{tree}(s)$ is an ultimately periodic word of the form uv^ω for some finite $u, v \in \Sigma^*$.

1.1.4 The Model-Checking Problem(s)

In the remainder of the course, we are going to define various logics with temporal models as models. Assume we are given a notion of *satisfaction* $\mathfrak{M} \models \varphi$ of formulæ from some logical language $\mathcal{L}$ by temporal models. We focus on either the linear time or the branching time behaviours of *finite* transition systems. Thus all our model-checking problems take as input a finite transition system $\mathfrak{S}$ and a specification as a formula φ of the logic $\mathcal{L}$.

We consider four questions on such inputs:

existential linear time model-checking written $\mathrm{LMC}^{\exists}(\mathcal{L})$: does $w \models \varphi$ for some $w \in \mathrm{Runs}(\mathfrak{S})$?

universal linear time model-checking written $\mathrm{LMC}^{\forall}(\mathcal{L})$: does $w \models \varphi$ for all $w \in \mathrm{Runs}(\mathfrak{S})$?

existential branching time model-checking written $\mathrm{BMC}^{\exists}(\mathcal{L})$: does $t \models \varphi$ for some $t \in \mathrm{Trees}(\mathfrak{S})$?

universal branching time model-checking written $\mathrm{BMC}^{\forall}(\mathcal{L})$: does $t \models \varphi$ for all $t \in \mathrm{Trees}(\mathfrak{S})$?

Since our logics are usually closed under negation, observe that the existential and universal variants are complement of each other. In the case of branching time, if $|I| \leq 1$, then the existential and universal problems coincide—this is not true for linear time behaviours, why? Finally, in the case of paths, as discussed in Remark 1.4, the linear time and branching time variants are equivalent.

Complexity-wise, we will always assume that the input transition system is represented 'efficiently,' using for instance a sparse adjacency matrix, so that the size $|\mathfrak{S}|$ of $\mathfrak{S}$ will be defined as $|S| + |T|$. The size $|\varphi|$ of the formula φ is simply the number of nodes in the syntactic tree of φ. We may sometimes call the complexity of the above decision problem the **combined complexity** of the model-checking problem.

In many examples, specifications might be rather simple, and we are more interested in the question of how a model-checking algorithm scales to large systems. If we fix the formula φ, we obtain a different decision problem for every φ. We will say that a model-checking problem of a logic $\mathcal{L}$ is in a complexity class C in **data complexity** if the problem is in C for every formula φ of $\mathcal{L}$. The model-checking problem is C-complete in data complexity if it is in C in data complexity, and there is a formula φ such that it is C-hard.

1.2 First-Order Logic

We consider the logic $\mathrm{FO}(\mathrm{AP}, <)$, whose formulas are defined by the following grammar:

$$\varphi ::= p(x) \mid x = y \mid x < y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi$$

where p ranges over AP and x, y are variables from some countable set Var.

A formula is *closed* if every variable is bound to an existential quantifier. We freely use the derived connectives $\wedge, \forall, \ldots$ as usual. We want to interpret formulas over temporal models, and for this we need to see them as structures over which we can interprete the symbols $p \in \mathrm{AP}$, $=$, and $<$. This is done in the obvious way by considering the ordered set of points.

Let $\mathfrak{M} = (\mathbb{T}, <, h)$ be a temporal model, φ a $\mathrm{FO}(\mathrm{AP}, <)$ formula, and $\nu\colon \mathrm{Var} \to \mathbb{T}$ be an assignment of first-order variables to points. The relation $\mathfrak{M} \models_\nu \varphi$ is defined inductively by

$$\begin{aligned}
&\mathfrak{M} \models_\nu p(x) && \text{iff } \nu(x) \in h(p)\ ,\\
&\mathfrak{M} \models_\nu x = y && \text{iff } \nu(x) = \nu(y)\ ,\\
&\mathfrak{M} \models_\nu x < y && \text{iff } \nu(x) < \nu(y)\ ,\\
&\mathfrak{M} \models_\nu \neg\varphi && \text{iff } \mathfrak{M} \not\models_\nu \varphi\ ,\\
&\mathfrak{M} \models_\nu \varphi \vee \psi && \text{iff } \mathfrak{M} \models_\nu \varphi \text{ or } \mathfrak{M} \models_\nu \psi\ ,\\
&\mathfrak{M} \models_\nu \exists x \,.\, \varphi && \text{iff } \exists i \in \mathbb{T} \,.\, \mathfrak{M} \models_{\nu[x\mapsto i]} \varphi\ .
\end{aligned}$$

where $\nu[x \mapsto i]$ maps x to i and $y \neq x$ to $\nu(y)$.

A temporal model satisfies a closed formula φ, written $\mathfrak{M} \models \varphi$, if $\mathfrak{M} \models \varphi$ for some/any assignment ν.

For various reasons, FO is not a logic of choice for specification. First, FO formulæ can quickly become difficult to write and to read. Second, the complexity of decision problems rises very quickly: the various model-checking problems introduced in Section 1.1.4 are all TOWER-complete (Stockmeyer, 1974) in combined complexity, i.e. require time (or space) resources bounded by a tower of exponentials whose height depends on the size of the input.

1.3 Modal Logic

As a preliminary to the temporal logics we will study in this course, we shall start with a very simple modal logic ML. The formulæ of ML are defined by the abstract syntax

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond\varphi$$

where p ranges over AP. Note that, unlike in FO, there are no explicit variables.

The semantics in terms of a temporal model $\mathfrak{M} = (\mathbb{T}, <, h)$ are relative to a time point i in $\mathbb{T}$: the model satisfies φ in i, written $\mathfrak{M}, i \models \varphi$, in the following inductive cases:

$$\begin{aligned}
\mathfrak{M}, i &\models p && \text{iff } i \in h(p)\ ,\\
\mathfrak{M}, i &\models \neg\varphi && \text{iff } \mathfrak{M}, i \not\models \varphi\ ,\\
\mathfrak{M}, i &\models \varphi \vee \psi && \text{iff } \mathfrak{M}, i \models \varphi \text{ or } \mathfrak{M}, i \models \psi\ ,\\
\mathfrak{M}, i &\models \Diamond\varphi && \text{iff } \exists i' \in \mathbb{T}\ .\ i < i' \text{ and } \mathfrak{M}, i' \models \varphi\ .
\end{aligned}$$

The intention is for $\Diamond\varphi$ to implicitly quantify on some $<$-related point where φ should hold. Further logical connectives can be defined as usual, e.g. $\top \overset{\text{def}}{=} p \vee \neg p$, $\varphi \wedge \psi \overset{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$, etc. More interestingly, the dual $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$ allows to quantify universally: $\mathfrak{M}, i \models \Box\varphi$ iff for all $i' \in \mathbb{T}$, if $i < i'$, then $\mathfrak{M}, i' \models \varphi$.

Over pointed temporal frames, i.e. temporal frames with a distinguished point i_0 (like 0 in $(\mathbb{N}, <)$ or the root of a tree), we write $\mathfrak{M} \models \varphi$ whenever $\mathfrak{M}, i_0 \models \varphi$.

Example 1.5. Let us consider a specification for a server answering some requests. As in Example 1.2, we set $\text{AP} \overset{\text{def}}{=} \{a, r, h\}$.

Let us first work with universal linear time model checking. We can express the following properties: the server never halts:

$$\Box\neg h \tag{1.6}$$

in all runs, the server always receives at least one request:

$$\Diamond r \tag{1.7}$$

in all runs, infinitely many acknowledgements are sent:

$$\Box\Diamond a \tag{1.8}$$

in all runs, every request is eventually followed by an acknowledgement:

$$\Box(r \implies \Diamond a)\ . \tag{1.9}$$

Observe that the transition system of Figure 1.1 satisfies formulæ (1.6) and (1.7), but violates (1.8) and (1.9): (1.4) is a counter-model for both.

When moving our attention to branching time model-checking, some unexpected effect occurs. Indeed, the tree in Figure 1.2 satisfies all of (1.6–1.9): at every position where r holds, just choose the upward branch to find an a immediately after. Note that the lowest branch in that tree yields the counter-model run (1.4), but somehow the proposed formulæ fail to forbid this run when using branching time semantics.

1.3.1 Standard Translation into First-Order

For every ML formula φ, there is a $\mathrm{FO}(\mathrm{AP},<)$ formula $\mathrm{ST}_x(\varphi)$ with a single free variable x, such that over all temporal models $\mathfrak{M}$ and points i

$$\mathfrak{M}, i \models \varphi \text{ iff } \mathfrak{M} \models_{\nu[x\mapsto i]} \mathrm{ST}_x(\varphi) \tag{1.10}$$

for some/any assignment ν. Indeed, it suffices to implement the Tarski-style semantics of ML into first-order logic:

$$\begin{aligned}
\mathrm{ST}_x(p) &\overset{\text{def}}{=} p(x) \;, \\
\mathrm{ST}_x(\neg\varphi) &\overset{\text{def}}{=} \neg\mathrm{ST}_x(\varphi) \;, \\
\mathrm{ST}_x(\varphi \vee \psi) &\overset{\text{def}}{=} \mathrm{ST}_x(\varphi) \vee \mathrm{ST}_x(\psi) \;, \\
\mathrm{ST}_x(\Diamond\varphi) &\overset{\text{def}}{=} \exists y. x < y \wedge \mathrm{ST}_y(\varphi) \;.
\end{aligned}$$

Exercise 1.1 (Standard Translation to FO)**.** Translate the formulæ from Example 1.5 into equivalent first-order formulæ.

The standard translation can be performed in logarithmic space, and entails the decidability of all the model-checking problems for ML. However, as we are going to see in the next two subsections, the general complexity bounds for FO are much higher than those we can establish when restricting our attention to ML.

Exercise 1.2 (Transitive Frames)**.** Show that the following formulæ φ are **valid** over all time flows, i.e. that, for every temporal model $\mathfrak{M}$ and point i, $\mathfrak{M}, i \models \varphi$:

$$\Box(p \implies q) \implies (\Box p \implies \Box q) \tag{\textbf{K}}$$

$$\Diamond\Diamond p \implies \Diamond p \tag{4}$$

1.3.2 Model-Checking Branching Behaviours

Let us focus for a moment on the model-checking problems for ML over branching time behaviours of finite transition systems.

Direct Semantics

The first remark is that the semantics of ML can be reformulated *directly* in terms of the transition system $\mathfrak{S} = (S, T, I, \mathrm{AP}, \ell)$ and states $s \in S$:

$$\begin{aligned}
&\mathfrak{S}, s \models p && \text{iff } p \in \ell(s) \\
&\mathfrak{S}, s \models \neg\varphi && \text{iff } \mathfrak{S}, s \not\models \varphi \\
&\mathfrak{S}, s \models \varphi \vee \psi && \text{iff } \mathfrak{S}, s \models \varphi \text{ or } \mathfrak{S}, s \models \psi \\
&\mathfrak{S}, s \models \Diamond\varphi && \text{iff } \exists s'.(s, s') \in T^+ \text{ and } \mathfrak{S}, s' \models \varphi
\end{aligned}$$

where T^+ denotes the transitive closure of T.

One can indeed check by induction on φ that

$$\mathfrak{S}, s \models \varphi \text{ iff } \mathrm{tree}(s), s \models \varphi \ . \tag{1.11}$$

The non-trivial case of the induction occurs for formulæ $\Diamond\varphi$: by definition, $\mathrm{tree}(s), s \models \Diamond\varphi$ iff there exists a point $ss_1 \cdots s_n$ such that $\mathrm{tree}(s), ss_1 \cdots s_n \models \varphi$. However, the latter satisfaction relation holds iff $\mathrm{tree}(s_n), s_n \models \varphi$ (because the subtree at $ss_1 \cdots s_n$ of $\mathrm{tree}(s)$ is isomorphic to $\mathrm{tree}(s_n)$), which by induction hypothesis is iff $\mathfrak{S}, s_n \models \varphi$.

Complexity

The main result pertaining to the branching time model-checking problems is their tractability:

Theorem 1.6. *BMC*$^\exists$(ML) *and BMC*$^\forall$(ML) *are* P*-complete.*

The P-easiness part of the statement was observed by Clarke et al. (1986) (and generalised by Arnold and Crubille (1988) to the *alternation-free* fragment of the *modal μ-calculus*):

Proposition 1.7. *The branching-time model-checking problems for finite transition systems $\mathfrak{S}$ and ML formulæ φ can be answered in $O(|\mathfrak{S}| \cdot |\varphi|)$.*

Proof. Define for a transition system $\mathfrak{S} = (S, T, I, \mathrm{AP}, \ell)$ the **satisfaction set** of a ML formula φ as the subset of S that satisfies φ, i.e. as

$$[\![\varphi]\!]_{\mathfrak{S}} \stackrel{\text{def}}{=} \{s \in S \mid \mathfrak{S}, s \models \varphi\} \ . \tag{1.12}$$

Then existential model-checking reduces to checking $[\![\varphi]\!]_{\mathfrak{S}} \cap I \neq \emptyset$ and universal model-checking to checking $I \subseteq [\![\varphi]\!]_{\mathfrak{S}}$.

An $O(|\mathfrak{S}| \cdot |\varphi|)$ algorithm can then compute $[\![\varphi]\!]_{\mathfrak{S}}$ by induction on φ, where each step can be performed in $O(|\mathfrak{S}|)$:

$$\begin{aligned}
[\![p]\!]_{\mathfrak{S}} &= \{s \in S \mid p \in \ell(s)\} \ , \\
[\![\neg\varphi]\!]_{\mathfrak{S}} &= S \setminus [\![\varphi]\!]_{\mathfrak{S}} \ , \\
[\![\varphi \vee \psi]\!]_{\mathfrak{S}} &= [\![\varphi]\!]_{\mathfrak{S}} \cup [\![\psi]\!]_{\mathfrak{S}} \ , \\
[\![\Diamond\varphi]\!]_{\mathfrak{S}} &= (T^{-1})^+\big([\![\varphi]\!]_{\mathfrak{S}}\big) \ .
\end{aligned}$$

This last computation of the transitive pre-image of $[\![\varphi]\!]_{\mathfrak{S}}$ by T can use for instance a depth-first search algorithm. □

The P-hardness of BMC$^\exists$(ML) entails however that the problem is unlikely to be efficiently parallelisable (see Schnoebelen, 2003, Theorem 3.8):

Proposition 1.8. *BMC*$^\exists$(ML) *and BMC*$^\forall$(ML) *are* P*-hard.*

Proof. The proof exhibits a reduction in logarithmic space from the *circuit value* problem to BMC$^\exists$(ML). In fact, we reduce from a retricted variant of the circuit value problem, where the circuit is *monotone* (only 'and' and 'or' gates), *alternating* (along every branch of the circuit, we alternate between 'and' and 'or' gates), and *synchronised* (the circuit is organised into layers where each gate leads to gates of the next layer and all the branches have the same length); this variant is still

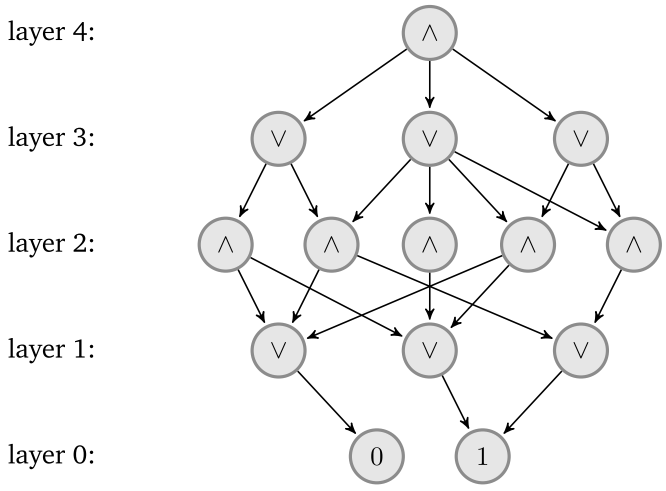

Figure 1.3: A monotone alternating synchronised Boolean circuit.

P-complete (Greenlaw et al., 1995, Theorem 6.2.5). We illustrate the reduction on the particular instance of Figure 1.3.

Note that any Boolean circuit C can be seen as a finite transition system $\mathfrak{S}_C$ with the gates as states and the input gate as initial state. Furthermore, we use $\mathrm{AP} \overset{\text{def}}{=} \{1\}$ and label the 1 exit gate by it, and all the other states by the empty set.

The additional restriction to monotone alternating synchronous circuits allows us to compute the value of the circuit in ML. The formula only depends on the number of layers and on the labelling (by 'and' or 'or') of the entry gate; in the case of Figure 1.3, the value of C is 1 if and only if

$$\mathfrak{S}_C \models \Box\Diamond\Box\Diamond 1\ . \qquad \Box$$

1.3.3 Model-Checking Linear Behaviours

Let us turn now to linear time behaviours. As mentioned earlier, linear time behaviours can be irregular, and as a consequence, the model-checking problems tend to be harder:

Theorem 1.9. *LMC*$^{\exists}(\mathrm{ML})$ *is* NP*-complete and LMC*$^{\forall}(\mathrm{ML})$ *is* coNP*-complete.*

Proof of NP*-easiness.* We shall not develop this proof in full. The crux of the argument is a *small model property* of ML over linear temporal models: if a formula φ is satisfied by some infinite word in Σ^ω, then there exists an ultimately periodic word uv^ω where u and v are finite words in Σ^* of polynomial length in $|\varphi|$ (Ono and Nakamura, 1980; Sistla and Clarke, 1985). This can be refined into non-deterministically finding a ultimately periodic word in $\mathrm{Runs}(\mathfrak{S})$ with polynomial representation, and then checking in polynomial time using Remark 1.4 and Proposition 1.7 that this word is a model of φ. $\Box$

Proof of NP*-hardness.* We reduce from the propositional satisfiability problem 3SAT. Given a 3SAT instance, i.e. a propositional formula $\Psi = \bigwedge_{1\leq i\leq m} \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$ over the set of variables $\{x_1,\ldots,x_n\}$, where the literals ℓ_{i_j} are either positive 'x_k' or negative '$\neg x_k$', we construct a finite transition system $\mathfrak{S}_n$ and a ML formula φ_Ψ such that there exists a run in $\mathrm{Runs}(\mathfrak{S}_n)$ that satisfies φ_Ψ iff Ψ is satisfiable.

The transition system $\mathfrak{S}_n$ over $\mathrm{AP} \overset{\text{def}}{=} \{x_1,\bar{x}_1,\ldots,x_n,\bar{x}_n\}$ is depicted in Figure 1.4. Any run over $\mathfrak{S}_n$ describes an assignment of the variables $x_k \in \{x_1,\ldots,x_n\}$, to true if we go through the state with label $\{x_k\}$ and to false if we go through the state with label $\{\bar{x}_k\}$. The purpose of the formula φ_Ψ is to check whether the

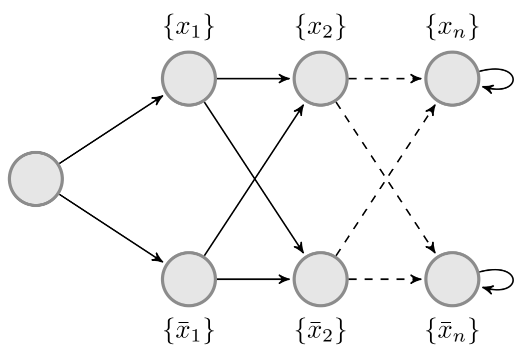

Figure 1.4: The transition system $\mathfrak{S}_n$. The runs in $\mathrm{Runs}(\mathfrak{S}_n)$ are in bijection with assignments of $\{x_1, \ldots, x_n\}$.

assignment described by the run in $\mathfrak{S}_n$ satisfies Ψ. We define therefore

$$\varphi_\Psi \overset{\text{def}}{=} \bigwedge_{1 \le i \le m} \Diamond \ell'_{i_1} \vee \Diamond \ell'_{i_2} \vee \Diamond \ell'_{i_3}$$

where $\ell'_{i_j} \overset{\text{def}}{=} x_k$ if $\ell_{i_j} = x_k$ and $\ell'_{i_j} \overset{\text{def}}{=} \bar{x}_k$ if $\ell_{i_j} = \neg x_k$. □

Exercise 1.3 (Trichotomous Frames)**.** A time flow $(\mathbb{T}, <)$ is **linear** if the relation $<$ is total, i.e. if for all points $i \neq j$, $i < j$ or $j < i$; a temporal model $\mathfrak{M} = (\mathbb{T}, <, h)$ is then linear if $(\mathbb{T}, <)$ is linear.

Show that the following formula is valid over all linear temporal models at every point:

$$\Diamond p \wedge \Diamond q \implies \Diamond(p \wedge \Diamond q) \vee \Diamond(p \wedge q) \vee \Diamond(q \wedge \Diamond p) \qquad \textbf{(.3)}$$

Conversely, consider a time flow $(\mathbb{T}, <)$ and a point i, and assume $\mathfrak{M}, i \models$ **.3** for all temporal models over $(\mathbb{T}, <)$. Consider the **generated subframe** $\mathbb{T}_i$ at i, i.e. consider the restriction of $\mathbb{T}$ to the points j such that $i < j$. Show that $\mathbb{T}_i$ is linear.

Chapter 2

References

Arnold, A. and Crubille, P., 1988. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29(2):57–66. doi:10.1016/0020-0190(88)90029-4. Cited on page 9.

Baier, C. and Katoen, J.P., 2008. *Principles of Model Checking*. MIT Press.

Clarke, E.M., Emerson, E.A., and Sistla, A.P., 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263. doi:10.1145/5397.5399. Cited on page 9.

Diekert, V. and Gastin, P., 2008. First-order definable languages. In Flum, J., Grädel, E., and Wilke, T., editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press. http://dare.uva.nl/document/154959#page=262.

Greenlaw, R., Hoover, H.J., and Ruzzo, W.L., 1995. *Limits to Parallel Computation:* P-*Completeness Theory*. Oxford University Press. Cited on page 10.

Ono, H. and Nakamura, A., 1980. On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica*, 39(4):325–333. doi:10.1007/BF00713542. Cited on page 10.

Schnoebelen, Ph., 2003. The complexity of temporal logic model checking. In Balbiani, Ph., Suzuki, N.Y., Wolter, F., and Zakharyaschev, M., editors, *AiML'02), 4th Workshop on Advances in Modal Logics*, pages 393–436. King's College Publication. http://www.lsv.fr/Publis/PAPERS/PDF/Sch-aiml02.pdf. Cited on page 9.

Sistla, A.P. and Clarke, E.M., 1985. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749. doi:10.1145/3828.3837. Cited on page 10.

Stockmeyer, L.J., 1974. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT, Department of Electrical Engineering, MIT. Cited on page 7.

van Glabbeek, R.J., 2001. The linear time—branching time spectrum I. the semantics of concrete, sequential processes. In Bergstra, J.A., Ponse, A., and Smolka, S.A., editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier. doi:10.1016/B978-044482830-9/50019-9. Cited on page 5.