Minimisation of Models Satisfying CTL Formulas

Serenella Cerrito
IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France
serenella.cerrito@univ-evry.fr

Amélie David
IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France
amely.david@laposte.net

Valentin Goranko
Stockholm University, Sweden
University of Johannesburg (visiting professorship), South Africa
valentin.goranko@philosophy.su.se

Abstract

We study the problem of minimisation of a given finite pointed Kripke model satisfying a given CTL formula, with the only objective to preserve the satisfaction of that formula in the resulting reduced model. We consider minimisations of the model with respect both to state-based redundancies and formula-based redundancies in that model. We develop a procedure computing all such minimisations, illustrate it with some examples, and provide some complexity analysis for it.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Computing methodologies → Temporal reasoning

Keywords and phrases CTL, model minimisation, bisimulation reduction, tableaux-based reduction

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.13

Funding The work of Valentin Goranko was supported by a research grant 2015-04388 of the Swedish Research Council.

1 Introduction

1.1 The problem of study and our proposal

The Computation Tree Logic CTL ([7], [9]) is one of the most useful and applicable temporal logics in computer science, because of its good balance between expressiveness and computational efficiency of model checking. One of the main problems that arise in its practical use is the *state explosion problem*, which calls for methods for reducing the size of the state transition systems arising when modelling real programs or systems. A lot of research has been done over the past three-four decades in addressing and resolving that problem by applying various techniques, such as bisimulation minimisations, abstraction refinements, BDD-based symbolic representations and symbolic model checking, partial order reductions, SAT-based model checking, etc. (cf [8] for comprehensive and up-to-date accounts of these). Most of these techniques follow the idea of applying minimisations, reductions, or abstractions to the original model, prior to doing model checking of the desired properties in it, by ensuring that the reduced model preserves all relevant properties (e.g., by being bisimulation equivalent to the original one). This approach is certainly very natural and has proved to be practically very useful.

Here, however, we take a somewhat different approach, viz. we study the problem of minimisation of a given finite pointed Kripke model (aka, pointed interpreted transition system) $(\mathcal{M}, s)$ *that is already known to satisfy a given* CTL *formula* θ, with the only objective to preserve the satisfaction of that formula in the minimised model. We argue that this problem is natural and important, too, because the formula θ can be viewed as

a formal specification of all critical features that the system must possess. Then, one may naturally want to synthesise a smallest and simplest possible abstract model of the system that satisfies that specification at its initial state, e.g. in order to facilitate further multiple verifications of various other properties and eventually its practical implementation. For instance, such formulas might be specifications of components of a product transition system, and such product constructions usually produce large redundancies that should preferably be eliminated before the actual implementation.

The main problem of this study is more precisely described as follows. We assume that some pointed Kripke model $(\mathcal{M}, s)$, satisfying a given CTL formula θ is already available, e.g. extracted from a real system or constructed by some of the well-known methods (tableaux, automata, etc, see e.g. [10]). We are then interested in producing a "minimal" such pointed model out of the given one, that still satisfies θ. By "minimal" here we mean a pointed model that cannot be further reduced by means of general and explicitly specified reducing operations, such as identifying states or taking submodels, to an even smaller one that still satisfies θ. We note that a given model satisfying a given formula may not be minimal with respect to that property for at least two different reasons: it may have redundancies caused by bisimilar states, and it may have redundancies with respect to the formula that it must satisfy. Thus, minimizing procedures for both types of redundancies are generally necessary, because most of the currently used methods for constructing satisfying models of CTL formulas (typically, tableaux or automata-based) do not usually produce minimal models in either sense.

Contributions. Our main contribution is the development of a minimization procedure that eliminates both kinds of redundancies. Respectively, our proposal, in a nutshell, is to combine and iterate two reduction procedures:

- ▷ ***Bisimulation reduction procedure,*** based on some of the well-known algorithms, e.g. in [16] or [13]. This procedure eliminates redundancies caused by bisimilar states and works in low polynomial (at most quadratic) time. Note that for our purpose we are only interested in bisimulation reduction with respect to the language of the given formula θ (called θ-bisimilarity in the following).
- ▷ ***Formula-driven reduction procedure,*** based on a tableaux-like construction. It implements two simple minimisation ideas:
  - to satisfy a disjunction, use part of the model to satisfy just one disjunct;
  - select only minimal (irreducible) sets of necessary successors of each state.

Because of the possible choices in both cases above, this procedure branches and eventually may produce several minimisations.

While this work focuses on minimisation of models of CTL formulas, we also consider in passing the simpler case of minimisation of models of formulas of the basic modal logic.

Related work. As the problem is important and very natural, there is much related work, though, up to our knowledge, none of it addresses exactly the same problem or follows the same approach as ours. We give a brief (and, for lack of space, quite incomplete) overview of related approaches to model minimisation, in a roughly chronological order.

Algorithmic bisimulation minimisation of Kripke models (aka, interpreted transition systems) has been explored extensively in the literature, going back to [13] and [16]; see [14] for an overview and references therein. The question of generation of minimal models with respect to bisimulation has been studied e.g. in [3], [2]. In [12] a method is proposed for obtaining a minimal transition system, representing a communicating system given by a set of parallel processes. More related to our work are [6] and [17], which explore *compositional*

minimization. There, a system on which a CTL formula θ needs to be model-checked is taken to be the product of n transitions systems $M_1, .., M_n$. A local model-checking of each M_i allows for the computation of a BDD representing a reduced number of transitions, so to reduce the final global product. Unlike our work, however, bisimulation-based reductions are not taken into account and redundancies caused by disjunctions are not considered. The above approach is then extended in [1], where a notion of formula-dependent state equivalence is proposed. However, again, redundancies caused by disjunctions are ignored, as well as subset inclusion (see Section 3.4).

Mogavero and Murano [15] have proposed a logic extending CTL* and internalizing minimal model construction by means of two minimal model quantifiers, Λ and Ξ. That approach, while thematically closely related, is somewhat orthogonal and incomparable to ours. The main difference is that we do not extend the CTL language to reason about truth in minimal models of formulas, but are interested in the *actual computing* of the minimizations of a model with respect to a formula, which we do purely semantically and constructively. Besides, we consider a stronger notion of minimality, taking into account also bisimulation. Thus, the objectives, approaches and results are quite different. We compare the two approaches with some more details and an example in Section 5.

Bozzelli and Pearce [4] explore the idea of "temporal equilibrium model" of an LTL formula, satisfying minimality requirement with respect to state labels. Cerrito and David [5] investigate the question of bisimulation minimisation of models of the multi-agent extension ATL of CTL.

Structure of the paper. We start with a brief background on the logic CTL and on bisimulation in Section 2. In Section 3 we describe two versions of our minimization procedure and we illustrate it on some examples. Some results about properties of the procedure are established in Section 4. We conclude by indicating some lines of future work in Section 5. A few proofs of auxiliary results are put in a short appendix.

2 Preliminaries

2.1 CTL: syntax and semantics

Here we only provide brief basic preliminaries on CTL. For further details see e.g. [10, Ch.7]. The syntax of CTL is given by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{E}(\varphi\,\mathsf{U}\,\varphi) \mid \mathsf{A}(\varphi\,\mathsf{U}\,\varphi)$$

where $\top$ is the logical constant for truth, Prop is a set of **proposition symbols** and $p \in$ Prop. We also use the following abbreviations: $\mathsf{AX}\,\varphi := \neg\mathsf{EX}\,\neg\varphi$, $\mathsf{EF}\,\varphi := \mathsf{E}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{AF}\,\varphi := \mathsf{A}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{EG}\,\varphi := \neg\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi := \neg\mathsf{EF}\,\neg\varphi$.

The set of atomic propositions occurring in a formula φ is denoted by $\mathrm{prop}(\varphi)$. The **basic modal logic BML** is the fragment of CTL that does not involve the operator U, i.e. extends propositional logic only with EX.

CTL formulas are interpreted over transition systems.

▶ **Definition 1.** *A **transition system** is a pair $\mathcal{T} = (S, R)$, where S is a nonempty set of **states** and $R \subseteq S \times S$ is a **transition relation** on S. Unless otherwise specified, transition systems will be assumed* serial *(this requirement is typically imposed for models of* CTL *but not for models of* BML*), i.e. for every $s \in S$ there is $s' \in S$ such that $(s, s') \in R$. When a distinguished state $s \in S$ is considered, $(\mathcal{T}, s)$ is called a **rooted (at s) transition system**, or a **pointed transition system**. A **path** in $\mathcal{T}$ is a sequence $\lambda : \mathbb{N} \to S$ such*

that $(\lambda(n), \lambda(n+1)) \in R$ *for every* $n \in \mathbb{N}$. *An* ***interpreted transition system*** *(ITS) over* $\mathcal{T}$ *is a tuple* $\mathcal{M} = (S, R, \mathrm{Prop}, L)$, *where* Prop *is a set of* ***proposition symbols*** *and* $L : S \to \mathcal{P}(\mathrm{Prop})$ *is a* ***state description function*** *defining for every state in* S *the set of atomic propositions true at that state. A* ***rooted (pointed) interpreted transition system*** $(\mathcal{M}, s)$ *is defined accordingly.*

Given an ITS $\mathcal{M} = (S, R, \mathrm{Prop}, L)$ *and any subset* P *of* Prop, *we define the* ***reduction of*** $\mathcal{M}$ ***to*** P *to be the ITS* $\mathcal{M}|_P = (S, R, P, L|_P)$, *where* $L|_P : S \to \mathcal{P}(P)$ *is defined by* $L|_P(s) = L(s) \cap P$ *for every* $s \in S$.

Given an ITS $\mathcal{M} = (S, R, \mathrm{Prop}, L)$, *an ITS* $\mathcal{M}' = (S', R', \mathrm{Prop}, L')$ *is said to be a* ***substructure*** *of* $\mathcal{M}$ *whenever* $S' \subseteq S$, L' *is the restriction of* L *to* S', *and* $R' = R \cap (S' \times S')$. *By an abuse of language, we say that* $\mathcal{M}'$ *is a substructure of* $\mathcal{M}$ *also when* $R' = (R \cap (S' \times S')) \cup \{< t_1, t_1 >, \ldots, < t_n, t_n >\}$ *where* $\{t_1, .., t_n\} \subseteq S'$, *for any* $n \geq 0$. *The ITS* $\mathcal{M}'$ *is said to be a* ***proper substructure*** *of* $\mathcal{M}$ *when* $S' \subset S$.

▶ **Definition 2.** *Let* $\mathcal{M} = (S, R, \mathrm{Prop}, L)$ *be an interpreted transition system,* $s \in S$ *and* φ *a* CTL*-formula.* ***Truth of*** φ ***at*** s ***in*** $\mathcal{M}$, *denoted by* $\mathcal{M}, s \models \varphi$, *is defined inductively on* φ *as follows (we give here only the non-boolean cases):*

- $\mathcal{M}, s \models \mathsf{EX}\,\varphi$ *iff there is a state* s' *such that* $(s, s') \in R$ *and* $\mathcal{M}, s' \models \varphi$.
- $\mathcal{M}, s \models \mathsf{E}(\varphi\,\mathsf{U}\,\psi)$ *iff there is a path* λ *in* $\mathcal{M}$ *starting from* s *and* $i \geq 0$ *such that* $\mathcal{M}, \lambda(i) \models \psi$ *and* $\mathcal{M}, \lambda(j) \models \varphi$ *for every* $j < i$.
- $\mathcal{M}, s \models \mathsf{A}(\varphi\,\mathsf{U}\,\psi)$ *iff for every path* λ *in* $\mathcal{M}$ *starting from* s, *there is* $i \geq 0$ *such that* $\mathcal{M}, \lambda(i) \models \psi$ *and* $\mathcal{M}, \lambda(j) \models \varphi$ *for every* $j < i$.

An ITS $(\mathcal{M}, s)$ *is a* ***pointed model of*** φ *whenever* $\mathcal{M}, s \models \varphi$.

2.2 Types, components, and extended closure of CTL formulas

We use some notions and terminology from the literature on tableaux-based satisfiability decision methods (see e.g. [10, Ch.13]). Formulas of CTL can be classified as: **literals**: $\top$, $\neg\top$, $p, \neg p$, where $p \in \mathrm{Prop}$, **successor formulas**: $\mathsf{EX}\,\varphi$ and $\neg\mathsf{EX}\,\varphi$, **conjunctive formulas** (also called ***α*-formulas**), and **disjunctive formulas** (also called ***β*-formulas**). The formulas in the last three classes have respective **components** that are given by Table 1. For convenience, the tables provide also the components of some defined formulas (e.g. $\mathsf{EF}\,\psi$). It is well-known (cf. [10, Ch.13]) that any conjunctive (resp. disjunctive) formula in the table is equivalent to the conjunction (resp. disjunction) of its components.

Table 1 Types of formulas and their components.

Conjunctive formula	Components
$\neg\neg\varphi$	φ
$\neg(\varphi \vee \psi)$	$\neg\varphi,\ \neg\psi$
$\neg\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\neg\psi,\ \neg\varphi \vee \neg\mathsf{EXE}(\varphi\,\mathsf{U}\,\psi)$
$\neg\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\neg\psi,\ \neg\varphi \vee \neg\mathsf{AXA}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{EG}\,\varphi$	$\varphi,\ \mathsf{EXEG}\,\varphi$
$\mathsf{AG}\,\varphi$	$\varphi,\ \mathsf{AXAG}\,\varphi$

Disjunctive formula	Components
$\varphi \vee \psi$	$\varphi,\ \psi$
$\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\psi,\ \varphi \wedge \mathsf{EXE}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\psi,\ \varphi \wedge \mathsf{AXA}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{EF}\,\psi$	$\psi,\ \mathsf{EXEF}\,\psi$
$\mathsf{AF}\,\psi$	$\psi,\ \mathsf{AXAF}\,\psi$

Successor formula	Components
$\mathsf{EX}\varphi$ (existential successor formula)	φ
$\neg\mathsf{EX}\varphi$ (universal successor formula)	$\neg\varphi$

▶ **Definition 3.** *The **extended closure** of a formula φ is the least set of formulas $\mathsf{ecl}(\varphi)$ such that:*

1. $\varphi \in \mathsf{ecl}(\varphi)$,
2. $\mathsf{ecl}(\varphi)$ *is closed under taking all components of each formula ψ in $\mathsf{ecl}(\varphi)$, i.e., conjunctive, disjunctive and successor components, according to the type of ψ*

For any set of formulas Γ we define $\mathsf{ecl}(\Gamma) := \bigcup\{\mathsf{ecl}(\varphi) \mid \varphi \in \Gamma\}$.

A formula $\mathsf{E}(\varphi \mathsf{U} \psi)$ (in particular, $\mathsf{EF}\,\psi$) is said to be an **existential eventuality** and $\mathsf{A}(\varphi \mathsf{U} \psi)$ (in particular, $\mathsf{AF}\,\psi$) – a **universal eventuality**.

2.3 Bisimulations and invariance

We recall here the well-known notion of bisimilarity of interpreted transition systems (see, for instance, [14] or [10, Ch.3]).

▶ **Definition 4.** *Let $\mathcal{M}_1 = (S_1, R_1, \mathrm{Prop}, L_1)$ and $\mathcal{M}_2 = (S_2, R_2, \mathrm{Prop}, L_2)$ be two interpreted transition systems over the same set of propositions* Prop. *A relation $\beta \subseteq S_1 \times S_2$ is a **bisimulation** between $\mathcal{M}_1$ and $\mathcal{M}_2$, denoted $\mathcal{M}_1 \overset{\beta}{\rightleftarrows} \mathcal{M}_2$, iff for all $s_1 \in S_1$ and $s_2 \in S_2$, $s_1 \beta s_2$ implies:*

1. *Atom Equivalence: $\mathsf{L}_1(s_1) = \mathsf{L}_2(s_2)$;*
2. *Forth condition: For any $r_1 \in S_1$, if $s_1 R_1 r_1$ then there is some $r_2 \in S_2$ such that $s_2 R_2 r_2$ and $r_1 \beta r_2$;*
3. *Back condition: For any $t_2 \in S_2$, if $s_2 R_2 t_2$ then there is some $t_1 \in S_1$ such that $s_1 R_1 t_1$ and $t_1 \beta t_2$.*

*Two states $s_1 \in S_1$ and $s_2 \in S_2$ are **bisimilar** if there is a bisimulation β between $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $s_1 \beta s_2$. We denote that by $(\mathcal{M}_1, s_1) \overset{\beta}{\rightleftarrows} (\mathcal{M}_2, s_2)$ (or, just $(\mathcal{M}_1, s_1) \rightleftarrows (\mathcal{M}_2, s_2)$ when β is inessential) and say that the rooted models $(\mathcal{M}_1, s_1)$ and $(\mathcal{M}_2, s_2)$ are **locally bisimilar**. If there is a bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ that links every state in S_1 to some state of S_2 and vice versa, we say that $\mathcal{M}_1$ and $\mathcal{M}_2$ are **(globally) bisimilar**.*

The following is a minor adaptation of a well-known result relating bisimulations and logic (see e.g. [18] or [10, Ch.3]). Here bisimulation is between reductions of ITS to a subset P of atomic propositions, thus *Atom Equivalence* is relativised to the propositions in P only.

▶ **Proposition 5** (Relativised bisimulation invariance)**.** *Let φ be a* CTL *formula,* $\mathrm{prop}(\varphi) \subseteq$ Prop, $\mathcal{M}_1$=(S_1,R_1, Prop, L_1) *and* $\mathcal{M}_2$=(S_2, R_2, Prop, L_2), *and $\beta \subseteq S_1 \times S_2$ be a local bisimulation between $(\mathcal{M}_1|_{\mathrm{prop}(\varphi)}, s_1)$ and $(\mathcal{M}_2|_{\mathrm{prop}(\varphi)}, s_2)$. Then $(\mathcal{M}_1|_{\mathrm{prop}(\varphi)}, s_1) \models \varphi$ iff $(\mathcal{M}_2|_{\mathrm{prop}(\varphi)}, s_2) \models \varphi$.*

When $\mathcal{M}_1 \overset{\beta}{\rightleftarrows} \mathcal{M}_2$ and $\mathcal{M}_1 = \mathcal{M}_2 = \mathcal{M}$ we say that **β is a bisimulation in $\mathcal{M}$**. Every such bisimulation is an equivalence relation in $\mathcal{M}$ and therefore generates a quotient-structure from $\mathcal{M}$ which we call the **quotient of $\mathcal{M}$ with respect to β**. It is well-known (see e.g. [11] or [10, Ch.3]) that amongst all bisimulations in $\mathcal{M}$ there is a largest one, $\beta_{\mathcal{M}}$. The quotient of $\mathcal{M}$ with respect to $\beta_{\mathcal{M}}$, hereafter denoted by $\widetilde{\mathcal{M}}$, is called the **bisimulation collapse of $\mathcal{M}$**. Note that every two different states in $\widetilde{\mathcal{M}}$ are non-bisimilar.

All these concepts relativise to reductions of ITS with respect to subsets P of atomic propositions. Note that, the smaller the subset P is, the larger the respective largest bisimulation in $\mathcal{M}|_P$, and therefore the smaller the bisimulation collapse $\widetilde{\mathcal{M}|_P}$. Therefore, when trying to minimize a model of a given formula θ with respect to bisimulations, we will be interested in $\widetilde{\mathcal{M}|_{\mathrm{prop}(\theta)}}$.

3 Model minimisation procedure (MMP)

3.1 Brief informal description

Our main aim is to develop an efficient procedure that minimises – in a sense to be made precise later – any given finite pointed model $(\mathcal{M}, s)$ of a CTL formula θ.

To facilitate and optimise that procedure, we precede it with global model checking in $\mathcal{M}$ of the formulas in the extended closure of θ. Since model checking of CTL formulas is very efficient, viz., bi-linear in both the size of the model and the length of the formula ([7], see also [10, Ch.7]), this preprocessing would not increase the overall complexity of the minimisation procedure.

Now, given $(\mathcal{M}, s)$ and the input formula θ, such that $(\mathcal{M}, s) \models \theta$, by applying global model checking in $\mathcal{M}$ we identify the set $\|\theta\|_{\mathcal{M}}$ of all states in $\mathcal{M}$ satisfying θ. If θ must be satisfied in *the same* (up to bisimulation collapse) state as s in the obtained minimal model, then the procedure works as described further shortly. If, however, satisfying θ at any state in the obtained minimal model will be sufficient for the purposes of the intended minimization, then a slightly different approach may be preferable: consider all states $t \in \|\theta\|_{\mathcal{M}}$, call the minimisation procedure to $(\mathcal{M}, t)$ for each of them, and finally select one of the obtained minimal models. Alternatively, to avoid some of that work, select amongst all states $t \in \|\theta\|_{\mathcal{M}}$ only those, for which the generated at t submodel of $\mathcal{M}$ is minimal by inclusion with respect to the others, and only apply the minimisation procedure to them.

We emphasize that either of these approaches may be preferable, depending on the concrete case. So, we are only listing them here as reasonable options, but the actual choice of concrete approach is left to the agent (or tool) performing the minimisation.

We assume hereafter that the possible selection of states indicated above has already been performed and the task now is to minimise a given pointed model $(\mathcal{M}, s)$ so that the formula θ is eventually satisfied at (the image of) the same state s in the minimised model.

As noted in the introduction, the minimisation procedure that we develop aims at detecting and eliminating two kinds of redundancies in $\mathcal{M}$, described below. These may have to be applied repeatedly, in an order discussed further, in Section 4.1.

1. *Model-based redundancies*, that arise when the model contains different states that are bisimilar with respect to the language of the input formula. These redundancies are eliminated by applying a well-known bisimulation minimisation procedure, after ignoring the atomic propositions not occurring in the formula. This procedure is deterministic and produces a unique (up to state renaming) reduced model – the bisimulation quotient. 5
2. *Formula-based redundancies*, that arise when the model contains "unnecessary" states, that can be removed without affecting the truth of the formula. Typically, such redundancies arise when:
   - **(i)** the model satisfies both disjunctive components of a disjunctive (sub)formula at some state, instead of only one of them, or
   - **(ii)** a state has more successors than what is needed to satisfy the (sub)formulas that have to be true there, or
   - **(iii)** a state is not reached in the process of the evaluation of the formula. These include all states that are not reachable by finite transition paths from the root state. In the case of a BML formula of modal depth $\leq n$ these are also all states not reachable in n transition steps from the root state.

These redundancies are eliminated by applying a tableaux-like procedure on the given input model, systematically selecting a single branch in the search / decision tree whenever a disjunction is to be satisfied, and selecting only a *minimal subset of necessary successors*

of each state added to the selection; this notion is precisely defined in Section 3.4. This procedure is non-deterministic and produces at least one, but possibly many reduced models, some of which may contain others. After its completion we also remove all obtained reduced models that are not minimal by inclusion.

Note that the preliminary global model checking is also useful in the tableaux-like minimisation procedure to select only minimal subsets of necessary successors of the current state, as well as to select in advance the shortest possible paths in the model realizing required eventualities. This will be illustrated on the running examples of minimising redundant models presented further.

3.2 Running examples

▶ **Example 6.** Consider the rooted model $(\mathcal{M}_1, s)$ shown in Figure 1 and the following formulas :

$\phi_1 = \mathsf{EX}\, p \wedge \mathsf{AF}\,(q \vee \mathsf{EF}\, p)$, $\phi_2 = \mathsf{EX}\, \neg p \wedge \mathsf{EX}\, q \wedge \mathsf{AG}\,(q \rightarrow p)$,
$\phi_3 = \mathsf{EX}\, \neg p \wedge \mathsf{EX}\, \mathsf{E}((p \wedge q)\, \mathsf{U}\, \neg q)$, $\phi_4 = \mathsf{EX}\, q \wedge \mathsf{EG}\,(\neg q \wedge p)$,
$\theta_1 = \phi_1 \vee \phi_2$, $\theta_2 = \phi_1 \vee \phi_3$, $\theta_3 = \phi_1 \vee \phi_4$.

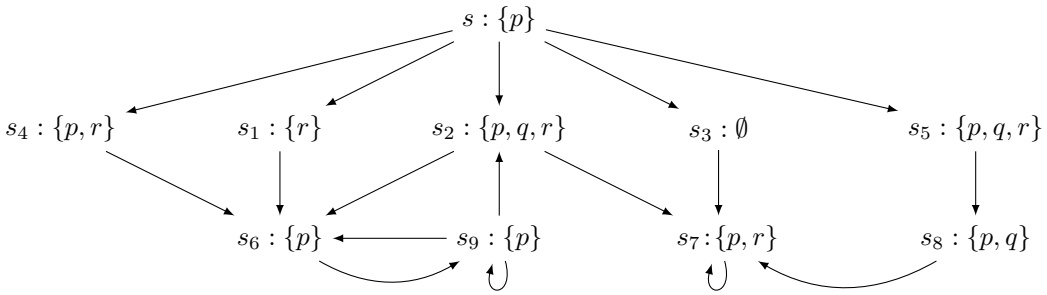

Figure 1 The model $\mathcal{M}_1$.

$\mathcal{M}_1$ satisfies at s all ϕ_i, for $i = 1..4$. Hence, it satisfies each of θ_1, θ_2 and θ_3 but, as we will show, it has unnecessarily many states.

▶ **Example 7.** Model $\mathcal{M}_2$ in Figure 4 satisfies $(\mathcal{M}_2, s) \models \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$. Again, we will show that it contains states that are unnecessary for that purpose.

3.3 Bisimulation reduction (BR)

As explained in Section 2.3, in our procedure of bisimulation minimization of a (pointed) model $(\mathcal{M}, s)$ satisfying a given CTL formula θ, in order to obtain a smallest possible bisimulation collapse of $\mathcal{M}$ that still satisfies θ we only need to compute the bisimulation collapse of *the reduction* $\mathcal{M}_\theta = \mathcal{M}|_{\mathrm{prop}(\theta)}$ *of* $\mathcal{M}$ *to the language of* θ. The resulting pointed ITS $(\widetilde{\mathcal{M}|_\theta}, \tilde{s})$ still satisfies θ and has the minimal number of states amongst all ITS that satisfy θ and are θ-bisimilar to $\mathcal{M}$. We call this formula-oriented procedure **θ-bisimulation minimisation of $\mathcal{M}$**.

Some essential remarks are in order.

(i) In order to preserve the satisfaction of θ it suffices to compute a *local* bisimulation collapse, of the submodel of $\mathcal{M}_\theta$ that is generated by s.

(ii) If θ is a BML-formula of modal depth n, then it suffices to compute the *n-bisimulation collapse of* $(\mathcal{M}_\theta, s)$, that identifies any two states satisfying the same formulas of depth up to n in the language of θ. That will, in general, produce an even smaller model.

(iii) The issue arises of what happens to the atomic propositions not occurring in θ. The procedure above ignores and forgets them completely. But that may be neither necessary nor desirable, even though we are currently only concerned with $\mathcal{M}$ as a model satisfying θ. This is because there may be other properties of $\mathcal{M}$, involving atoms not occurring in θ, the truth of which may be affected by the minimisation procedure and may be of importance later. So, we propose the following refinement: to keep a best possible record of the truth of each atom r not occurring in θ in the resulting reduced model $(\widetilde{\mathcal{M}|_\theta}, \tilde{s})$ by introducing, besides *true* and *false*, a third truth-value *both*, that will be assigned to r at each state in the collapsed model where original states with different truth values of r have been identified. Thus, the resulting refined model allows for 3-valued valuation of the truth of formulas involving such atomic propositions, that can be used for evaluating the truth of some formulas that contain them. We will not pursue systematically this idea here, but leave it to future work.

There are well-known efficient procedures for bisimulation minimisation based on partition refinement such as the Kanellakis-Smolka algorithm [13], optimized to the Paige-Tarjan algorithm in [16]. (For other, more involved and efficient algorithms see [14]; see also [1].) It is quite easy to refine most of these θ-bisimulation minimisation procedures to account for the refinements above, but for lack of space we will not spell out the details.

▶ **Example 8** (Example 6 continued)**.** Let us apply BR to the model $\mathcal{M}_1$ with respect to the language of the formulas θ_i of Example 6, i.e. over the set of atomic propositions $P = \{p, q\}$. The coarsest partition of the set of states corresponding to the maximal bisimulation relation in $\mathcal{M}_1|_P$ contains six clusters: $C_\square = \{s\}$, $C_\bullet = \{s_1, s_3\}$, $C_\circ = \{s_2, s_8\}$, $C_\times = \{s_4\}$, $C_\otimes = \{s_5\}$, $C_+ = \{s_6, s_7, s_9\}$. Note that, for instance, s_6 and s_9 are in the same cluster even if they do not agree on the valuation of the propositional letter r, as it does not belong to the language of our interest. These clusters of bisimilar states are visualized in Figure 2. The corresponding quotient model $\mathcal{M}'_1$, collapsing all states belonging to the same cluster into a unique state, is given in Figure 3.

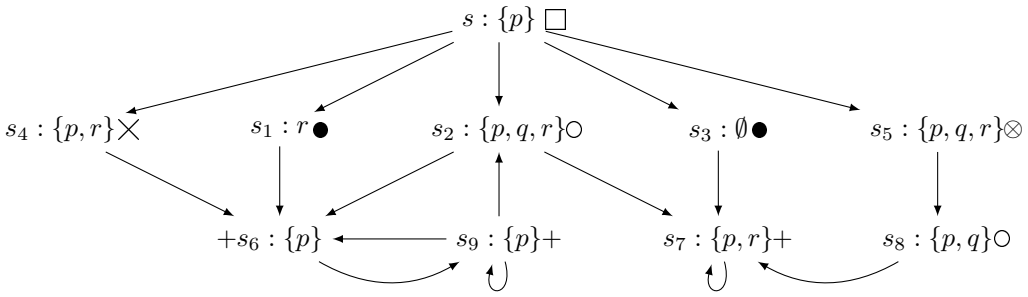

Figure 2 $\{p, q\}$-bisimilar states in the model $\mathcal{M}_1$.

3.4 Tableaux-based reduction (TR)

As explained earlier, the purpose of this reduction is to remove parts of the model that are unnecessary for satisfying the target input formula, typically when satisfying disjunctive choices and selecting successors. The input of the procedure TR is a pointed ITS $(\mathcal{M}, s)$

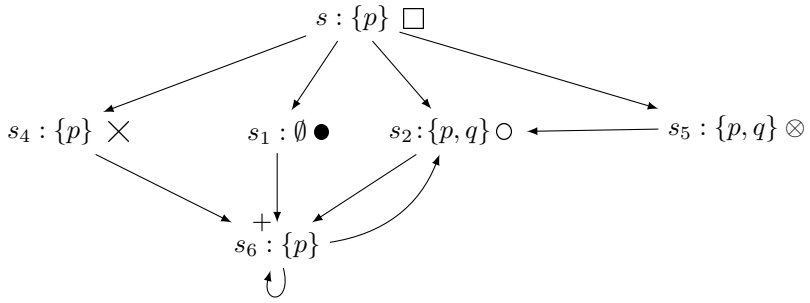

Figure 3 The bisimulation quotient model $\mathcal{M}'_1 = \widetilde{\mathcal{M}_1|_{\{p,q\}}}$

and a formula θ such that $M, s \models \theta$ is given/known to be true (our initial assumption). The output is a family of reduced pointed ITS $(\mathcal{M}_1, s), \ldots (\mathcal{M}_q, s)$ satisfying θ. Here is an informal outline of the overall procedure:

1. TR starts with a global model checking in $\mathcal{M}$ of the formulas in the extended closure $\mathsf{ecl}(\theta)$ of the input formula θ.
2. Then TR runs a tableau-like procedure that iteratively labels states of $\mathcal{M}$ with sets of formulas. At start, the root state s of M is labeled with $\{\theta\}$, while all other states have an empty label. Then labels are possibly modified repeatedly until stabilisation, according to a sub-procedure LAB that we outline later. A non-deterministic run of LAB produces a submodel $\mathcal{M}'$ of $\mathcal{M}$ with state space S' consisting of all states in S with non-empty labels.

   When all the possible runs of LAB are executed, in parallel or consecutively, a list of reduced pointed models $(\mathcal{M}_1, s), \ldots (\mathcal{M}_k, s)$ is produced.
3. Check for *subset inclusion*[1]: if $\mathcal{M}_i$ is included as a substructure in $\mathcal{M}_j$, then remove $\mathcal{M}_j$ from the list. The procedure eventually returns the family of minimal by inclusion reduced pointed models that remain in the list.

We are now going to describe more formally and precisely the procedure outlined above.

▶ **Definition 9.** *Let $(\mathcal{M}, s)$ be a pointed ITS and let Γ be a set of formulas that hold at s. A **(non-deterministic) optimal saturation** of Γ is a procedure OS that, when applied non-deterministically to Γ produces a set of formulas Δ such that $\Gamma \subseteq \Delta$ by repeatedly applying the following operations until saturation:*

1. *Initially, $\Delta := \Gamma$.*
2. *If a conjunctive formula φ is in Δ then OS adds both its components to Δ;*
3. *If a disjunctive formula φ is in Δ and none of its disjunctive components is in Δ, then OS chooses non-deterministically any of these components which is true at s and adds it to Δ. However, the following exception applies: if φ is an eventuality, i.e. $\mathsf{E}(\chi \,\mathsf{U}\, \psi)$, $\mathsf{EF}\, \psi$, $\mathsf{A}(\chi \,\mathsf{U}\, \psi)$, or $\mathsf{AF}\, \psi$, and none of its components is in Δ but ψ is true at s, then OS adds only ψ to Δ.*

*The sets Δ produced by runs of OS are called **(optimally) saturated extensions** of Γ. Γ is said to be **optimally saturated** if it equals an optimally saturated extension of itself.*

[1] More generally, TR can check for isomorphic embeddings, but that may increase substantially the complexity of the whole procedure.

The adjective "optimal" in the above definition is due to the third item, that minimizes the number of disjunctive components required to be true and aims at fulfilling eventualities as soon as possible. Note that if $\Gamma \subseteq \mathsf{ecl}(\theta)$ for a given formula θ and Δ is an optimally saturated extension of Γ, then $\Delta \subseteq \mathsf{ecl}(\theta)$. Moreover all the elements of Δ are true at s, by construction. In particular, so are all the successor formulas occurring in Δ.

▶ **Definition 10.** *Let $\mathcal{M}$ be an ITS, $s \in \mathcal{M}$, let Γ be an optimally saturated set of formulas true at s, and let $\Gamma_{suc} = \{\neg\mathsf{EX}\psi_1, ..., \neg\mathsf{EX}\psi_k, \mathsf{EX}\varphi_1, ..., \mathsf{EX}\varphi_m\}$ be its subset of successor formulas (where each of k and m can be 0). A **minimal set of successors of s** w.r.t. Γ_{suc} is a set U of states in $\mathcal{M}$ that are (immediate) successors of s and:*

1. *Each existential successor formula $\mathsf{EX}\varphi_j$ in Γ_{suc} has a "witness" in U, viz. some state $w(\varphi_j) \in U$ such that $\mathcal{M}, w(\varphi_j) \models \varphi_j$;*
2. *U is minimal with respect to the above property: if any state is removed from U then the resulting set S' lacks a witness for at least one $\mathsf{EX}\,\varphi_j \in \Gamma_{suc}$.*
3. *In case when $m = 0$, an arbitrary self-looping successor of s is added to U, just for the sake of seriality.*

By hypothesis, all formulas in Γ_{suc} are true at s. Therefore, for all $\neg\mathsf{EX}\,\psi_i \in \Gamma_{suc}$, the formula $\neg\psi_i$ is true at each state $s' \in U$.

The procedure ANALYSE given below takes as input an ITS, a state s in it, and a set of formulas $\mathcal{L}(s)$ currently labelling that state. It updates $\mathcal{L}(s)$ by saturating it and adding formulas to the current labels of some successors of s, to produce the updated labels as an output. The top procedure LAB calls ANALYSE.

The procedure ANALYSE**.**

1. Construct an optimal saturation Δ of $\mathcal{L}(s)$ and reset the value of $\mathcal{L}(s)$ to Δ.
2. If $\mathcal{L}(s)_{suc} = \{\neg\mathsf{EX}\psi_1, ..., \neg\mathsf{EX}\psi_k, \mathsf{EX}\varphi_1, ..., \mathsf{EX}\varphi_m\}$ is the subset of successor formulas of $\mathcal{L}(s)$, then build a minimal set U of successors of s w.r.t. $\mathcal{L}(s)_{suc}$.
3. For each $s' \in U$: if $s' = w(\varphi_j)$, then add φ_j and all $\neg\psi_i$, $1 \leq i \leq k$, to the current value of $\mathcal{L}(s')$ (if they are not already in it).

The procedure LAB**.**

1. Initialization: set s to be the current state, $\mathcal{L}(s) := \{\theta\}$ and $\mathcal{L}(s') := \emptyset$ for each other state of $\mathcal{M}$.
2. Until all labels $\mathcal{L}(s')$ of states s' of $\mathcal{M}$ become stable, do:
   a. Apply ANALYSE to the current state t.
   b. Then for each state t' in the minimal set of successors U of t produced by ANALYSE at t, set t' to be the current state and recursively apply ANALYSE there.

Note that, for the sake of simplicity, here we are giving the pseudo-code for a non-deterministic run of LAB. It can be converted to a deterministic algorithm, producing the entire family of reduced models, by using suitable bookkeeping and backtracking mechanisms.

▶ **Example 11** (Example 1 continued)**.** Let us apply LAB to the model $\mathcal{M}'_1$ of Figure 3 and the formula $\theta_1 = \phi_1 \vee \phi_2$ that holds at s. At the initialisation, $\mathcal{L}(s) = \{\theta_1\}$, while the labels of all other states are the empty set. Since both ϕ_1 and ϕ_2 are true at s, a non-deterministic run of LAB makes a choice of which of them to put in an optimized non-deterministic saturation of $\mathcal{L}(s)$. Consider two cases:

1. Suppose that the choice $\phi_1 = \mathsf{EX}\,p \wedge \mathsf{AF}\,(q \vee \mathsf{EF}\,p)$ is made. Then both conjunctive components of ϕ_1, $\mathsf{EX}\,p$ and $\mathsf{AF}\,(q \vee \mathsf{EF}\,p)$, are added to the saturation. The latter formula is an eventuality, whose disjunctive components are $q \vee \mathsf{EF}\,p$ and $\mathsf{AXAF}\,(q \vee \mathsf{EF}\,p)$. Here both components are true at s, but optimality forces us to choose $q \vee \mathsf{EF}\,p$. Now only $\mathsf{EF}\,p$ is true at s, so it is the chosen disjunctive component. In turn, $\mathsf{EF}\,p$ is an eventuality whose disjunctive components are p and $\mathsf{EXEF}\,p$. Since p is true at s then p is chosen. To summarise, the corresponding non-deterministic saturation of $\{\theta_1\}$ built here is the set $\{\theta_1,\ \phi_1,\ \mathsf{EX}\,p,\ \mathsf{AF}\,(q \vee \mathsf{EF}\,p),\ q \vee \mathsf{EF}\,p,\ \mathsf{EF}\,p,\ p\}$. It becomes the new value of $\mathcal{L}(s)$. Its set of successor formulas is $\{\mathsf{EX}\,p\}$, for which we obtain three minimal sets of successors of s, namely $\{s_2\}$, $\{s_4\}$ and $\{s_5\}$. A non-deterministic run of LAB chooses one of them, and adds the formula p to the corresponding state. In each of the three cases, the analysis of the newly labeled state produces no new label and the run halts, respectively producing: the sub-model $\mathcal{M}_a$ of $\mathcal{M}'_1$ containing just the states $\{s, s_2\}$, the sub-model $\mathcal{M}_b$ containing just the states $\{s, s_4\}$, and the sub-model $\mathcal{M}_c$ containing just the states $\{s, s_5\}$ (with loops, respectively, on s_2, s_4 and s_5).
2. Suppose now that the choice $\phi_2 = \mathsf{EX}\,\neg p \wedge \mathsf{EX}\,q \wedge \mathsf{AG}\,(q \to p)$ is made.
   Reasoning as above, by choosing suitable minimal sets of successors, we get:
   - either a candidate model having s, s_1 and s_2 as states, hence strictly including $\mathcal{M}_a$, and therefore excluded as a true minimal model by the inclusion-check that follows the application of LAB procedure in TR,
   - or, a candidate model that strictly includes $\mathcal{M}_c$ and is also disregarded.

   Hence, after the inclusion-check, the complete run of TR on $\mathcal{M}'_1$ produces the family of reduced models consisting of $\mathcal{M}_a$, $\mathcal{M}_b$ and $\mathcal{M}_c$.

▶ **Example 12** (Example 7 continued)**.** Consider the model $\mathcal{M}_2$ of Figure 4 that satisfies $\psi = \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$ at s. An application of the procedure BR w.r.t. the set of propositions $\{p, q\}$ identifies states s_1 and s_6 as bisimilar, producing the model $\mathcal{M}'_2$ described in Figure 4. Then, running TR on that model and ψ removes s_5 and produces the model $\mathcal{M}''_2$ described in Figure 4. The states s_3 and s_4 are now bisimilar, so a new application of BR to $\mathcal{M}''_2$ is necessary. It produces the model $\mathcal{M}^*_2$ of Figure 4, where s_3 and s_4 are now collapsed into one state. Such a model of $\mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$ cannot be further reduced. This example shows that *the procedure* BR *may have to be applied again after an application of* TR *in order to minimise further the model.*

4 Analysis and results

4.1 Minimisation procedures running BR and TR together

The examples run so far show that it may be necessary to alternate the procedures BR and TR in order to produce truly minimal models of the target formula. Indeed, none of the two procedures subsumes the other in terms of the outcomes. This can be seen by a simple example. Take, for instance, $\mathcal{M}$ to be the model $\mathcal{M}'_2$ of Figure 4 and $\psi = \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$, as in Example 12. If we run again BR on this input, we trivially get again $\mathcal{M}'_2$, since $\mathcal{M}'_2$ is already minimal with respect to ψ-bisimulation. However, running TR on $\mathcal{M}'_2$ and ψ produces the model $\mathcal{M}''_2$ shown in Figure 4. Thus, the two results are incomparable. More generally, observe also that both BR and TR are idempotent, i.e. neither of them produces new models if applied consecutively twice. These suggest that a minimising procedure might either start with BR and then alternate TR and BR phases (on the input produced by the previous phase) until stabilisation, or else start with TR and then alternate BR and TR phases until stabilisation. However we can bound the number of such alternations until stabilisation in both cases, due to the following result.

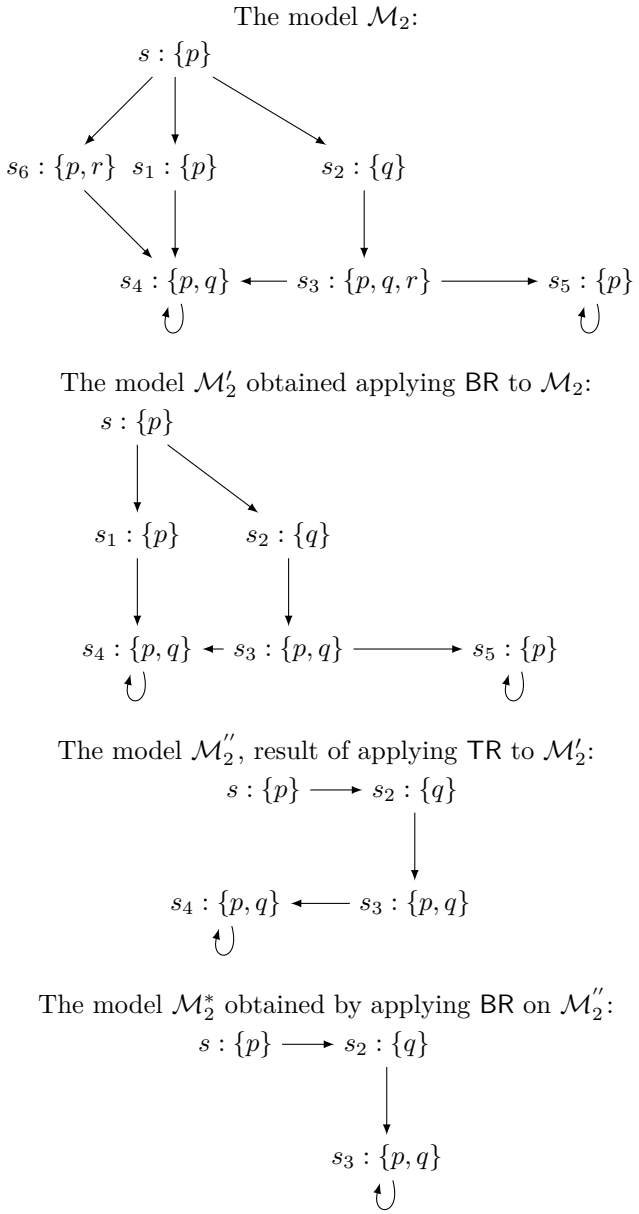

Figure 4 A complete reduction of the model $\mathcal{M}_2$.

▶ **Lemma 13.** *The reduction* TR *has to be applied only at most once, that is: given a pointed model* $(\mathcal{M}_1, t)$ *and a formula* θ*, let* $(\mathcal{M}_2, t)$ *be a reduced model produced by a run of* TR *on* $\mathcal{M}_1$ *and let* $(\mathcal{M}_3, \tilde{t})$ *be the result of running* BR *on* $(\mathcal{M}_2, t)$*. Then any run of* TR *on* $(\mathcal{M}_3, \tilde{t})$*,* θ *produces again* $(\mathcal{M}_3, \tilde{t})$ *as a result.*

Proof. Note that TR only removes states from its input model if they remain with empty labels. So, it suffices to observe that, if any formula $\phi \in \mathsf{ecl}(\theta)$ was added by the first run of TR to the label of a state $s \in \mathcal{M}_2$, then the same formula will be added to the label of the respective collapse state $\tilde{s} \in \mathcal{M}_3$ produced by applying BR to $\mathcal{M}_2$, and therefore that state will be preserved in the application of TR to $\mathcal{M}_3$. The proof can be done by tracing step by step the run of TR on $\mathcal{M}_1$ producing $\mathcal{M}_2$ and the respective run of TR on $\mathcal{M}_2 = \widetilde{\mathcal{M}_2}$. We omit the routine details. ◀

Therefore, there are only two different ways to organize the whole procedure:

MMP1: Start with TR, then apply BR to each obtained model.

MMP2: Start with BR, then apply TR to the obtained model, then again BR to each resulting model.

▶ **Example 14** (Example 12 continued)**.** In Example 12, we have actually run MMP2 on the model $\mathcal{M}_2$ (Figure 4) and the formula $\psi = \mathsf{EX}(\neg p \wedge \mathsf{EX}(p \wedge \mathsf{EX}(p \wedge q)))$. If we rather run MMP1 on the model $\mathcal{M}_2$, TR immediately produces the model $\mathcal{M}_2''$, then an application of BR to such a model makes s_3 and s_4 collapse and produces the minimal model $\mathcal{M}_2^*$.

4.2 Convergence and comparison of MM1 and MM2

▶ **Lemma 15.** *Given any pointed model $(\mathcal{M}, s)$ and a formula θ, every reduced pointed model produced from $(\mathcal{M}, s)$ by applying first* BR *and then* TR *can also be produced by applying first* TR *and then* BR*.*

Proof. Let $(\widetilde{\mathcal{M}}, \widetilde{s})$ be produced from $(\mathcal{M}, s)$ by applying BR and let $(\widetilde{\mathcal{M}}', \widetilde{s})$ be produced from $(\widetilde{\mathcal{M}}, \widetilde{s})$ by applying TR. It suffices to note that every run of procedure TR applied to $(\widetilde{\mathcal{M}}, \widetilde{s})$ to produce $(\widetilde{\mathcal{M}}', \widetilde{s})$ can be simulated, step by step, by a run of TR applied to $(\mathcal{M}, s)$, by selecting at every step a set of successors which are respectively θ-bisimulation equivalent to successors selected at the respective step of the run of TR applied to $(\widetilde{\mathcal{M}}, \widetilde{s})$. That would eventually produce a pointed model, on which BR would produce $(\widetilde{\mathcal{M}}', \widetilde{s})$. ◀

▶ **Theorem 16.** *For every initial pointed model $(\mathcal{M}, s)$ and a given formula φ:*

1. ***MMP1*** *and* ***MMP2*** *produce the same families of reduced models.*
2. *Every reduced pointed model produced by either of* ***MMP1*** *and* ***MMP2*** *is minimal in the following senses:*
   a. Bisimulation-minimal *with respect to the language of φ.*
   b. State-minimal*, in the sense that no state can be removed from $\mathcal{M}$ to still preserve the truth of φ at s.*

Proof. We first prove the second claim. The bisimulation-minimality is immediate, as both procedures end with BR. The state minimality follows from the minimality of every set of successors preserved by TR, and using Lemma 13.

Now, the first claim. First, every reduced pointed model produced by **MMP2** can also be produced by **MMP1**, by Lemma 15 and the idempotency of BR. For the converse inclusion, note that every run ρ of TR applied to a pointed model $(\mathcal{M}, s)$ and input formula θ can be lifted to a run $\widetilde{\rho}$ of TR on the θ-bisimulation quotient $(\widetilde{\mathcal{M}}, \widetilde{s})$ by selecting there the respective clusters of the selected successors in $(\mathcal{M}, s)$. Eventually, applying again BR to the resulting submodel $(\widetilde{\mathcal{M}}', \widetilde{s})$ would produce the same θ-bisimulation quotient as BR applied to the submodel of $(\mathcal{M}, s)$ produced by the run ρ of TR. ◀

We note that neither of the procedures **MMP1** and **MMP2** is intended, nor guaranteed, to produce a *smallest* possible model of the input formula, but only to minimise the input model in the senses described above. Indeed, e.g. the formula ψ in Example 12 has a smaller model than the model $\mathcal{M}_2^*$ in f Figure 4 that was obtained from $\mathcal{M}_2$ by the reduction procedure: a model with just two states, s, having label $\{p\}$, and its looping successor s_2 having label $\{p, q\}$.

We end this section with some complexity analysis. First, note that, despite the equivalence, the procedures **MMP1** and **MMP2** may have quite different performances. For instance, the deterministic version of **MMP1** can take in some cases an exponentially larger number of steps than **MMP2**, as shown by the following example.

▶ **Example 17.** Let θ be a formula of the form $\mathsf{EX}...\mathsf{EX}\,p$, where EX occurs n times, and let $\mathcal{M}$ be a pointed model that is a fully balanced binary tree of height n, satisfying p at each leaf and where all the states at the same level are bisimilar. Note that **MMP2**, starting with BR, will collapse all branches into one, and then TR will not make any change. On the other hand, **MMP1**, starting with TR, will produce 2^n isomorphic reduced models, each of them being a branch in the original model, i.e. a linear chain of length n. After checking for isomorphisms at the end, TR will leave just one of them, which BR will not change.

Now, to analyse the complexity, we can focus on the procedure **MMP1**, taking as inputs a formula θ and a pointed model $(\mathcal{M}, s)$, and returning a set[2] of minimal reduced models. **MMP1** first computes $\mathsf{ecl}(\theta)$ and does global model checking of all formulas in it in $\mathcal{M}$, in time linear in both $|\theta|$ (the size of θ) and $|\mathcal{M}|$ (the size of $\mathcal{M}$). Then, a non-deterministic run of the sub-procedure LAB in the worst case treats all formulas in $\mathsf{ecl}(\theta)$ and visits all the states in $\mathcal{M}$. Thus, it runs in time polynomial in $|\theta|$ and $|\mathcal{M}|$. Eventually, it produces a family of (possibly exponentially many, as evident from Example 17) minimal submodels, but for the sake of comparing and selecting the smallest of them, they can be produced consecutively, thus reusing space. Thus, TR can produce its output consecutively, in PSPACE. Bisimulation reduction of each of the models obtained by TR can be done in $O(m\ log\ n)$[16], where m is the number of transitions and n is the number of states of the model. Thus, finally, it takes polynomial space to produce every reduced pointed model consecutively, as an output of **MMP1**. A similar complexity analysis applies to **MMP2**.

5 Further work and concluding remarks

We have proposed a formula-oriented minimization procedure in two versions, **MMP1** and **MMP2**, that reduces the number of states of a model M satisfying a given CTL formula θ, by taking into account both possible θ-bisimulation redundancies as well as redundancies induced by the structure of θ. Using a tableau-like procedure for handling the second type of redundancies and combining the two kinds of reduction procedures are the main original ideas of our contribution. As already observed in the literature, to reduce the size of components with respect to their corresponding specification formulas can help to tackle the space explosion problems of product transition systems.

Our approach is related to, but different from, [15], as mentioned in the introduction. Not only we do not modify CTL syntax, but our notion of minimality is different and we solve a different algorithmic problem, too. Indeed, a formula $\phi_1 \Xi \phi_2$ in [15] holds at a state s of a model $\mathcal{M}$ when there is a minimal (and conservative, as defined in that work) *sub-structure* of $\mathcal{M}$ verifying ϕ_2 at s that verifies also ϕ_1. Here, minimality is with respect to an ordering of sub-structures of $\mathcal{M}$. In our case, minimisation includes also bisimulation reduction. Thus, for instance, consider again the rooted model $(\mathcal{M}_1, s)$ and the formula θ_1 of Example 6 and let θ_1' be $\theta_1 \wedge \mathsf{EX}\,\mathsf{EX}\,(p \wedge \neg q)$. Then running BR produces the quotient model $\mathcal{M}_1'$ exhibited by Figure 3, then a run of TR gives the model whose states are s, s_5, s_6 (with s connected to s_5, s_5 connected to s_6 and a loop on s_6). The latter is *not* a sub-structure of $\mathcal{M}_1$, and model-checking the formula $\top \Xi \theta_1'$ of the logic in [15] cannot produce it.

Future work includes extending our approach to model minimization to richer logics, in particular to the multi-agent extension ATL of CTL, whose models are minimized only with respect to (alternating) bisimulation in [5]. We also intend to implement **MMP1** and **MMP2** and to test experimentally and compare their performance in practical cases.

[2] Thus, the minimisation problem that this procedure solves is not a decision problem.

References

1 Adnan Aziz, Thomas Shiple, Vigyan Singhal, Robert Brayton, and Alberto Sangiovanni-Vincentelli. Formula-Dependent Equivalence for Compositional CTL Model Checking. *Formal Methods in System Design*, 21(2):193–224, 2002.

2 A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18(3):247–269, 1992.

3 Ahmed Bouajjani, Jean-Claude Fernandez, and Nicolas Halbwachs. Minimal Model Generation. In *Proc of CAV '90*, pages 197–203, 1990.

4 Laura Bozzelli and David Pearce. On the Expressiveness of Temporal Equilibrium Logic. In *Proc. of JELIA 2016*, pages 159–173, 2016.

5 Serenella Cerrito and Amélie David. Minimisation of ATL* Models. In *Proc. of TABLEAUX 2017*, pages 193–208, 2017.

6 Massimiliano Chiodo, Thomas R. Shiple, Alberto L. Sangiovanni-Vincentelli, and Robert K. Brayton. Automatic compositional minimization in CTL model checking. In *Proc. of ICCAD'1992*, pages 172–178, 1992.

7 E. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logics of Programs*, pages 52–71. Springer, 1981.

8 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, 2018. doi:10.1007/978-3-319-10575-8.

9 E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

10 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*, volume 58 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, October 2016. URL: http://www.cambridge.org/9781107028364.

11 V. Goranko and M. Otto. Model Theory of Modal Logic. In *Handbook of Modal Logic*, pages 249–330. Elsevier, 2007.

12 Susanne Graf and Bernhard Steffen. Compositional minimization of finite state systems. In Edmund M. Clarke and Robert P. Kurshan, editors, *Computer-Aided Verification*, pages 186–196, Berlin, Heidelberg, 1991. Springer.

13 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

14 Aceto L., Ingolfsdottir, and Jiri S. The Algorithmics of Bisimilarity. In Sangiorgi D. and Rutten J., editors, *Advanced topics in bisimulation and coinduction*, pages 100–171. Cambridge University Press, 2012.

15 Fabio Mogavero and Aniello Murano. Branching-Time Temporal Logics with Minimal Model Quantifiers. In *Developments in Language Theory, 13th International Conference, DLT 2009, Stuttgart, Germany, June 30 - July 3, 2009. Proceedings*, pages 396–409, 2009. doi:10.1007/978-3-642-02737-6_32.

16 R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

17 Thomas R. Shiple, Massimiliano Chiodo, Alberto L. Sangiovanni-Vincentelli, and Robert K. Brayton. Automatic Reduction in CTL Compositional Model Checking. In *Proc. of CAV'92*, pages 234–247, 1992.

18 Colin Stirling. Bisimulation and Logic. In Sangiorgi D. and Rutten J., editors, *Advanced topics in bisimulation and coinduction*, pages 173–195. Cambridge University Press, 2012.