Model Checking for Combined Logics

Massimo Franceschet[1] Angelo Montanari[1] Maarten de Rijke[2]

[1] Dip. di Matematica e Informatica, Università di Udine, Via delle Scienze 206
33100 Udine, Italy. E-mail: {francesc|montana}@dimi.uniud.it
[2] ILLC, University of Amsterdam, Pl. Muidergracht 24
1018 TV Amsterdam, The Netherlands. E-mail: mdr@wins.uva.nl

Abstract

We consider combined model checking procedures for the three ways of combining logics: temporalizations, independent combinations, and the join. We present results on computational complexity and report on experiments with implementations. We also discuss the relevance to time granular logics.

Key words: temporal logic, model checking, combining logics, time granularity.

1 Introduction

Concerns about modularity and the wish to join together different kinds of information have inspired various combinations of logics. As any interesting real world system is a complex entity, decomposing its descriptive and inferential requirements for design, verification, or maintenance purposes into simpler reasoning tasks is often the only plausible way forward [9]. Assuming that we have methods and tools available to tackle restricted tasks, how do we combine them to solve complex tasks. How do we combine them in such a way that features of the components are inherited by the combination? This question is known as the *transfer problem* [2]. Whether properties transfer from the components to the combination depends on the amount of interaction between the component logics; even in the presence of very weak forms of interaction (such as shared symbols), transfer may fail [11]. In the absence of interaction between the component logics, we often have transfer; such positive results are usually based on a *divide and conquer* strategy: split problems into sub-problems and delegate these to the components [4, 12].

From a *computational* point of view, the natural question in the setting of combining logics is: does it work? Can we *re-use* tools and procedures in a modular fashion? So far, most of the work towards answering this question has gone into putting together deductive engines. While there are no uniform solutions, there are many successful instances of combined proof procedures, especially for modal and modal-like logics [1]; these are often based on calculi satisfying special criteria or on translating the component logics into a background logic.

In this paper we study the combination of model checking procedures. In addition to the issues mentioned above, the direct motivation for this work has been the need to develop model checking procedures for granular logics [13, 14]. Such logics are able to model and reason about time at different grain levels, for instance, at the level of seconds and of micro-seconds. Instead of developing model checking procedures for granular logics from scratch, we want to synthesize them from existing (non-granular) ones. In Section 7 we will see that this is indeed possible. More generally, in contrast to combining deductive engines, combinations of model checking procedures are well behaved, even in the presence of interaction; indeed, this supports the general believe that modularity is easier to achieve in model checking than in theorem-proving approaches [10].

We start by recalling basic definitions in Section 2 and presenting three modes of combining logics in Section 3: temporalization, independent combinations, and the join. In Section 4, we consider combined model checking procedures for these three combinations. Section 5 contains results on computational complexity, and Section 6 reports on our experiments with implementations. In Section 7 we sketch the application of our ideas to granular logics, and we conclude in Section 8.

2 Temporal Logics

Let $\mathbf{L}$ be a logic system. We use $\mathcal{L}_{\mathbf{L}}$ and $\mathsf{K}_{\mathbf{L}}$ to denote the language and the set of models of $\mathbf{L}$, respectively. The language of a temporal logic $\mathbf{L}$ is based on a set $\mathcal{P}$ of proposition letters and extends that of propositional logic with a set of *temporal operators*. We write $OP(\mathbf{L})$ to denote the set of temporal operators of $\mathcal{L}_{\mathbf{L}}$.

Definition 2.1 (Syntax) The language $\mathcal{L}_{\mathrm{SUL}}$ of *Since and Until Logic* (SUL) is the smallest set X of formulas generated by the following rules: any proposition letter $p \in \mathcal{P}$ is in X; if ϕ, ψ are in X, then so are $\phi \wedge \psi$ and $\neg\phi$; and if ϕ, ψ are in X, then so are $\phi\mathbf{S}\psi$ and $\phi\mathbf{U}\psi$.

The language $\mathcal{L}_{\mathrm{CTL}^*}$ of the *computation tree logic* CTL^* has state formulas and path formulas. *State formulas* are obtained as follows: any proposition letter $p \in \mathcal{P}$ is a state formula; if ϕ, ψ are state formulas, then so are $\phi \wedge \psi$ and $\neg\phi$; if ϕ is a path formula, then $\mathbf{E}\phi$ and $\mathbf{A}\phi$ are state formulas. *Path formulas* are obtained as follows: every state formula is a path formula; if ϕ, ψ are path formulas, then so are $\phi \wedge \psi$ and $\neg\phi$; if ϕ, ψ are path formulas, then so is $\phi\mathbf{U}\psi$.

The language $\mathcal{L}_{\mathrm{CTL}}$ of the computation tree logic CTL is the set of *state formulas* generated by the rules for state formulas given before plus the following rule for path formulas: if ϕ, ψ are state formulas, then $\phi\mathbf{U}\psi$ is a path formula.

Definition 2.2 (Semantics) A *frame* for a temporal logic T is a pair $(W, \mathcal{R})$, where W is a set of *worlds*, or *states*, and $\mathcal{R}$ is a set of *accessibility relations* on W. We restrict ourselves to *binary* relations on W. A *model* for T is a triple $(W, \mathcal{R}, V)$, where $(W, \mathcal{R})$ is a frame for T and $V : W \rightarrow 2^{\mathcal{P}}$ is a *valuation function* mapping states into sets of proposition letters.

A frame for SUL, CTL, or CTL^* is a pair $\mathcal{F} = (W, \{R\})$, or simply (W, R), where W is a set of states and $R \subseteq W \times W$ is a binary relation on W. A *path* π in $\mathcal{F}$ is an infinite sequence of states s_0, s_1, … such that, for every $i \geq 0$, $R(s_i, s_{i+1})$; we write $\pi(j)$ for the element in the j-th position on the path π.

Truth of a SUL-formula ϕ in a model $\mathcal{M} = (W, R, V)$ with respect to a state $s \in W$ is defined as usual: for $p \in \mathcal{P}$, $\mathcal{M}, s \models_{\mathrm{SUL}} p$ if $p \in V(s)$, and $\mathcal{M}, s \models_{\mathrm{SUL}} \phi\mathbf{U}\psi$ iff $\mathcal{M}, t \models_{\mathrm{SUL}} \psi$ for some t such that Rst, while $\mathcal{M}, r \models_{\mathrm{SUL}} \phi$ for every r such that Rsr and Rrt; similarly, $\mathcal{M}, s \models_{\mathrm{SUL}} \phi\mathbf{S}\psi$ iff $\mathcal{M}, t \models_{\mathrm{SUL}} \psi$ for some t such that Rts, while $\mathcal{M}, r \models_{\mathrm{SUL}} \phi$ for every r such that Rtr and Rrs.

As to CTL^*-formulas, state formulas ψ are evaluated at a state s (notation: $\mathcal{M}, s \models_{\mathrm{CTL}^*} \psi$); path formulas ϕ are evaluated with respect to a path $\pi = s_0, s_1, \ldots$ and a position $j \in \mathbb{N}$ in π (notation: $\mathcal{M}, \pi, j \models_{\mathrm{CTL}^*} \phi$). For proposition letters $p \in \mathcal{P}$, we put $\mathcal{M}, s \models_{\mathrm{CTL}^*} p$ if $p \in V(s)$. For any path formula ϕ, we have that $\mathcal{M}, s \models_{\mathrm{CTL}^*} \mathbf{E}\phi$ if there is a path π starting at s such that $\mathcal{M}, \pi, 0 \models_{\mathrm{CTL}^*} \phi$; and $\mathcal{M}, s \models_{\mathrm{CTL}^*} \mathbf{A}\phi$ if for every path π starting at s we have $\mathcal{M}, \pi, 0 \models_{\mathrm{CTL}^*} \phi$. For a state formula ψ, we put $\mathcal{M}, \pi, j \models_{\mathrm{CTL}^*} \psi$ if $\mathcal{M}, \pi(j) \models_{\mathrm{CTL}^*} \psi$. Finally, for a pair of path formulas ϕ and ψ, we have that $\mathcal{M}, \pi, j \models_{\mathrm{CTL}^*} \phi\mathbf{U}\psi$ if $\mathcal{M}, \pi, i \models_{\mathrm{CTL}^*} \psi$ for some $i > j$ and $\mathcal{M}, \pi, k \models_{\mathrm{CTL}^*} \phi$ for every $j < k < i$ ($j \leq k < i$ if we adopted a non-strict version of $\mathbf{U}$).

The truth definition for CTL-formulas (notation: $\models_{\mathrm{CTL}}$) can be obtained by restricting $\models_{\mathrm{CTL}^*}$ to CTL-formulas.

3 Combining Logics

How do we combine logics? While many ways of combining logics have been explored, we restrict ourselves to only three of them: temporalization, independent combination, and the join. These three are certainly among the most popular and the ones that have been studied most extensively [4, 12, 5, 6].

§**3.1 Temporalization.** This is the simplest of the three modes of combining logics that we will consider; here, the two component languages are only allowed to interact in a very restricted way. More specifically, let $\mathbf{T}$ be a temporal logic and $\mathbf{L}$ an arbitrary logic. For simplicity we constrain $\mathbf{L}$ to be an extension of classical logic. We partition the set of $\mathbf{L}$-formulas into *boolean combinations* $BC_{\mathbf{L}}$ and *monolithic formulas* $ML_{\mathbf{L}}$: α belongs to $BC_{\mathbf{L}}$ if its outermost operator is a boolean connective; otherwise it belongs to $ML_{\mathbf{L}}$. We assume that $OP(\mathbf{T}) \cap OP(\mathbf{L}) = \emptyset$. The *combined language* $\mathcal{L}_{\mathbf{T}(\mathbf{L})}$ of the *temporalization* $\mathbf{T}(\mathbf{L})$ of $\mathbf{L}$ by means of $\mathbf{T}$ over the set of proposition letters $\mathcal{P}$ is obtained by replacing the atomic formation rule of $\mathcal{L}_{\mathbf{T}}$ (i.e., every proposition letter is a formula) by the following rule: every monolithic formula $\alpha \in \mathcal{L}_{\mathbf{L}}$ is an $\mathcal{L}_{\mathbf{T}(\mathbf{L})}$-formula.

A *model* for $\mathbf{T}(\mathbf{L})$ is a triple $(W, \mathcal{R}, g)$, where $(W, \mathcal{R})$ is a frame for $\mathbf{T}$ and g a total function mapping states in W to models in $\mathsf{K}_{\mathbf{L}}$. Given a model $\mathcal{M} = (W, \mathcal{R}, g)$ and a state $w \in W$, the semantics of the combined logic $\mathbf{T}(\mathbf{L})$ is obtained by replacing the usual semantic clause for atomic formulas of $\mathcal{L}_{\mathbf{T}}$ by the following clause: for all $\alpha \in ML_{\mathbf{L}}$, $\mathcal{M}, w \models_{\mathbf{T}(\mathbf{L})} \alpha$ if and only if $g(w) \models_{\mathbf{L}} \alpha$.

§**3.2 Independent Combination.** The independent combination of two logics puts together all the expressive power of the two component logics in an unrestricted way; our definitions are straightforward extensions of definitions found in [4, 12, 5]. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be two temporal logics defined over the same set of proposition letters $\mathcal{P}$, with $OP(\mathbf{T}_1) \cap OP(\mathbf{T}_2) = \emptyset$. The *fully combined language* $\mathcal{L}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$ of the *independent combination* $\mathbf{T}_1 \oplus \mathbf{T}_2$ over $\mathcal{P}$ is obtained by taking all connectives of $\mathbf{T}_1$ and $\mathbf{T}_2$, and the union of their formation rules.

To define the semantics of $\mathbf{T}_1 \oplus \mathbf{T}_2$, we need the following notion. Given a binary relation R, we write R^* for its transitive closure, and R^{-1} for its converse. Let $(W, \mathcal{R})$ be a frame. A *connected component* $(W', \mathcal{R}')$ of $(W, \mathcal{R})$ is a frame with (1) $\emptyset \neq W' \subseteq W$ and $\mathcal{R}' = \{R|_{W'} \mid R \in \mathcal{R}\}$; and (2) $(W', \mathcal{R}')$ is *connected*, i.e., for every u and v in W', with $u \neq v$, we have $(u, v) \in [\bigcup\{(R \cup R^{-1}) \mid R \in \mathcal{R}\}]^*$; and (3) $(W', \mathcal{R}')$ is *maximal*, i.e., there is no connected component $(W'', \mathcal{R}'')$ with $W' \subset W''$. Notice that an *isolated* point is a connected component.

A *model* for the combined logic $\mathbf{T}_1 \oplus \mathbf{T}_2$ is a 4-tuple $(W, \mathcal{R}_1, \mathcal{R}_2, V)$, where the connected components of $(W, \mathcal{R}_1, V)$ are in $\mathsf{K}_{\mathbf{T}_1}$, the connected components of $(W, \mathcal{R}_2, V)$ are frames underlying models in $\mathsf{K}_{\mathbf{T}_2}$, and W is the (not necessarily disjoint) union of the sets of states that constitute each connected component. Finally, $V : W \to 2^{\mathcal{P}}$ is a valuation function. The *truth definition* of the combined logic $\mathbf{T}_1 \oplus \mathbf{T}_2$ is obtained by taking the union of the semantic clauses for $\mathbf{T}_1$ and $\mathbf{T}_2$.

§**3.3 The Join.** Formulas in the language of the independent combination of two logics are evaluated at a single node in a model. The *join* introduces a separate dimension for each of the component logics, and we are allowed to express relations between the two dimensions. For notational simplicity we assume that our component logics are one-dimensional, i.e., evaluated at a single node only. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be two temporal logics. The *join* $\mathbf{T}_1 \otimes \mathbf{T}_2$ of $\mathbf{T}_1$ and $\mathbf{T}_2$ is obtained as follows. The *language* $\mathcal{L}_{\mathbf{T}_1 \otimes \mathbf{T}_2}$ of the logic system $\mathbf{T}_1 \otimes \mathbf{T}_2$ is the fully combined language of $\mathbf{T}_1$ and $\mathbf{T}_2$.

```
Function MC$_{\mathbf{T}(\mathbf{L})}$
Input: a $\mathbf{T}(\mathbf{L})$-model $\mathcal{M} = (W, \mathcal{R}, g)$ and a formula $\psi \in \mathcal{L}_{\mathbf{T}(\mathbf{L})}$

compute $\mathrm{MML}_{\mathbf{L}}(\psi)$ and abs$(\psi)$
for every $\alpha \in \mathrm{MML}_{\mathbf{L}}(\psi)$
    for every $w \in W$
        if MC$_{\mathbf{L}}(g(w), \alpha)$ = True then
          $V(w) = V(w) \cup \{p_\alpha\}$
return MC$_{\mathbf{T}}((W, \mathcal{R}, V)$, abs$(\psi))$
```

Figure 4.1: Model checking for temporalized logics.

A *model* for $\mathbf{T}_1 \otimes \mathbf{T}_2$ is a 5-tuple $(W_1, \mathcal{R}_1, W_2, \mathcal{R}_2, V)$, where $(W_1, \mathcal{R}_1)$ is a $\mathbf{T}_1$-frame and $(W_2, \mathcal{R}_2)$ is a $\mathbf{T}_2$-frame, and $V : W_1 \times W_2 \to 2^{\mathcal{P}}$ is a valuation mapping pairs of states to sets of proposition letters. Truth of a formula ϕ in a model $\mathcal{M} = (W_1, \mathcal{R}_1, W_2, \mathcal{R}_2, V)$, at states $s_1 \in W_1$ and $s_2 \in W_2$, is defined as follows. If $\phi = p$ ($p \in \mathcal{P}$), $\phi = (\phi_1 \wedge \phi_2)$, or $\phi = \neg\phi_1$, then $\mathcal{M}, s_1, s_2 \models_{\mathbf{T}_1 \otimes \mathbf{T}_2} \phi$ is defined as usual. If $\phi = \mathbf{O}(\phi_1, \ldots, \phi_n)$, with $\mathbf{O} \in OP(\mathbf{T}_i)$, we define $\mathcal{M}, s_1, s_2 \models_{\mathbf{T}_1 \otimes \mathbf{T}_2} \phi$ by replacing every occurrence of $\mathcal{M}, x$ in the definition of $\mathcal{M}, s_i \models_{\mathbf{T}_i} \phi$ ($i \in \{1, 2\}$) by $\mathcal{M}, x, s_2$ (if $i = 1$) or $\mathcal{M}, s_1, x$ (if $i = 2$).

In our presentation of the join of logics, we have followed [6]; in [8] a slightly different but equivalent construction is studied: the *product* of modal logics.

4 Model Checking for Combined Logics

In this section we consider model checking procedures for each of the modes of combining logics considered in Section 3.

§**4.1 Temporalization.** We first define the global model checking problem for the combined logic $\mathbf{T}(\mathbf{L})$; then, we give a general algorithm that solves it. Let $\mathcal{M} = (W, \mathcal{R}, g)$ be a $\mathbf{T}(\mathbf{L})$-model. We say that $\mathcal{M}$ is *finite* if W and $\mathcal{R}$ are finite and, for every $w \in W$, $g(w)$ is finite. Let $\mathcal{M} = (W, \mathcal{R}, g)$ be a *finite* $\mathbf{T}(\mathbf{L})$-model, $w \in W$ a state, and ψ a formula in $\mathcal{L}_{\mathbf{T}(\mathbf{L})}$. We focus on the *global model checking problem* for $\mathbf{T}(\mathbf{L})$: is there a state $v \in W$ such that $\mathcal{M}, v \models_{\mathbf{T}(\mathbf{L})} \psi$? We use 'model checker' for a program that solves the global model checking problem.

Let ψ be a $\mathbf{T}(\mathbf{L})$-formula and $\mathrm{MML}_{\mathbf{L}}(\psi)$ the set of *maximal* monolithic subformulas of ψ belonging to $\mathcal{L}_L$; abs(ψ) denotes the formula obtained from ψ by replacing every formula $\alpha \in \mathrm{MML}_{\mathbf{L}}(\psi)$ by the proposition letter p_α. Moreover, let MC$_{\mathbf{T}}$ and MC$_{\mathbf{L}}$ be model checkers for $\mathbf{T}$ and $\mathbf{L}$, respectively. Given an appropriate model checking instance, these programs return True if the corresponding instance is a "yes" instance, False otherwise. In Figure 4.1, we present the pseudo-code of a model checker MC$_{\mathbf{T}(\mathbf{L})}$ for $\mathbf{T}(\mathbf{L})$ that exploits MC$_{\mathbf{T}}$ and MC$_{\mathbf{L}}$. Let $\mathcal{M}$ be a finite model for $\mathbf{T}(\mathbf{L})$ and $\psi \in \mathcal{L}_{\mathbf{T}(\mathbf{L})}$. As a theorem, we have that if MC$_{\mathbf{L}}$ and MC$_{\mathbf{T}}$ are terminating, sound and complete, then, on input $\mathcal{M}$ and ψ, the function MC$_{\mathbf{T}(\mathbf{L})}$ terminates, returning either True or False. Moreover, if it returns True, then there exists $w \in W$ with $\mathcal{M}, w \models_{\mathbf{T}(\mathbf{L})} \psi$; and if it returns False, then, for every $w \in W$, $\mathcal{M}, w \not\models_{\mathbf{T}(\mathbf{L})} \psi$.

§**4.2 Independent Combination.** We now give a general algorithm for solving the global model checking problem for $\mathbf{T}_1 \oplus \mathbf{T}_2$. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be two temporal logics, and let $\mathcal{M} = (W, \mathcal{R}_1, \mathcal{R}_2, V)$ be a model for $\mathbf{T}_1 \oplus \mathbf{T}_2$. We say that $\mathcal{M}$ is *finite* if W, $\mathcal{R}_1$, and $\mathcal{R}_2$ are finite, and, for every $w \in W$, $V(w)$ is finite.

```
**Procedure** $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$
**Input**: a $\mathbf{T}_1 \oplus \mathbf{T}_2$-model $\mathcal{M} = (W, \mathcal{R}_1, \mathcal{R}_2, V)$ and a formula $\psi \in \mathcal{L}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$

**compute** $\mathcal{C}^1_{\mathcal{M}}$, $\mathcal{C}^2_{\mathcal{M}}$ and $MSub(\psi)$
**for every** $w \in W$ **let** $\overline{V}(w) = V(w)$
**for every** $i = 1, \ldots, |\psi|$
    **for every** $\phi \in MSub(\psi)$ such that $|\phi| = i$
        **case** on the form of $\phi$
            $\phi = p, p \in \mathcal{P}$: **skip**
            $\phi = \phi_1 \wedge \phi_2$: **for every** $w \in W$
                                **if** ($\phi_1 \in V(w)$ **and** $\phi_2 \in V(w)$) **then**
                                  $V(w) = V(w) \cup \{\phi\}$ ; $\overline{V}(w) = \overline{V}(w) \cup \{p_\phi\}$
            $\phi = \neg\phi_1$: **for every** $w \in W$
                          **if** (**not** $\phi_1 \in V(w)$) **then**
                            $V(w) = V(w) \cup \{\phi\}$ ; $\overline{V}(w) = \overline{V}(w) \cup \{p_\phi\}$
            $\phi = \mathbf{O}(\phi_1, \ldots, \phi_c)$, $\mathbf{O} \in OP(\mathcal{L}_{T_i})$, $i \in \{1, 2\}$
            **let** $\Phi = \{\alpha \in Sub(\phi) \cap MSub(\psi) \mid 1 < |\alpha| < |\phi|\}$ and $\phi' = \phi$
            **for every** $\alpha \in \Phi$ **replace** $\alpha$ in $\phi'$ with $p_\alpha$
            **for every** $(U, \mathcal{S}) \in \mathcal{C}^i_{\mathcal{M}}$
                **for every** $u \in U$ **let** $V'(u) = \overline{V}(u)$
                $\mathtt{MC}_{\mathbf{T}_i}((U, \mathcal{S}, V'), \phi')$
                **for every** $u \in U$
                    **if** $\phi' \in V'(u)$ **then**
                      $V(u) = V(u) \cup \{\phi\}$ ; $\overline{V}(u) = \overline{V}(u) \cup \{p_\phi\}$
```

Figure 4.2: Model checking independently combined logics.

The global model checking problem for $\mathbf{T}_1 \oplus \mathbf{T}_2$ is defined just as for $\mathbf{T}(\mathbf{L})$. $\mathcal{C}^1_{\mathcal{M}}$ and $\mathcal{C}^2_{\mathcal{M}}$ are the sets of connected components of $(W, \mathcal{R}_1)$ and $(W, \mathcal{R}_2)$, respectively. Since $\mathcal{M}$ is a model for $\mathbf{T}_1 \oplus \mathbf{T}_2$, every connected component in $\mathcal{C}^1_{\mathcal{M}}$ ($\mathcal{C}^2_{\mathcal{M}}$) is a model for $\mathbf{T}_1$ ($\mathbf{T}_2$). $Sub(\phi)$ is the set of subformulas of ϕ, and $MSub(\phi) \subseteq Sub(\phi)$ is constructed as follows. Let $S = Sub(\phi) \cap \mathcal{L}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$. Let $i \in \{1, 2\}$. For every formula $\mathbf{O}(\phi_1, \ldots, \phi_c)$ in S, with $\mathbf{O} \in OP(\mathcal{L}_{\mathbf{T}_i}) \cup \{\wedge, \vee, \neg\}$, if, for every $j = 1, \ldots, c$, ϕ_j is a proposition letter or its main operator is in $OP(\mathcal{L}_{\mathbf{T}_i}) \cup \{\wedge, \vee, \neg\}$, then delete formulas $\phi_1, \ldots, \phi_c$ from S; $MSub(\phi)$ is the set S at the end of this procedure. Note that if $\phi \in \mathcal{L}_{\mathbf{T}_i}$, then $MSub(\phi) = \{\phi\}$.

Below, we view model checkers as *procedures* that receive a model $(W, \mathcal{R}, V)$ and a formula ψ as input, and that extend the valuation V (which maps a state to a set of proposition letters) to a valuation V' mapping states to sets of *subformulas* of ψ in the following way: for every subformula ϕ of ψ and every node w, $V'(w)$ contains ϕ iff ϕ is true at w in $(W, \mathcal{R}, V)$. Let $\mathtt{MC}_{\mathbf{T}_1}$ and $\mathtt{MC}_{\mathbf{T}_2}$ be model checkers for $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively. In Figure 4.2, we present the pseudo-code of a model checker for $\mathbf{T}_1 \oplus \mathbf{T}_2$ that exploits the procedures $\mathtt{MC}_{\mathbf{T}_1}$ and $\mathtt{MC}_{\mathbf{T}_2}$. Let $\mathcal{M} = (W, \mathcal{R}_1, \mathcal{R}_2, V)$ be a finite model for $\mathbf{T}_1 \oplus \mathbf{T}_2$ and $\psi \in \mathcal{L}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$. As a theorem, we have that if $\mathtt{MC}_{\mathbf{T}_1}$ and $\mathtt{MC}_{\mathbf{T}_2}$ are terminating, sound, and complete, then, on input $\mathcal{M}$ and ψ, the procedure $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$ terminates. Moreover, if V' is the (extended) valuation function returned by $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$, then, for every subformula ϕ of ψ and every node $w \in W$, $\phi \in V'(w)$ iff $\mathcal{M}, w \models_{\mathbf{T}_1 \oplus \mathbf{T}_2} \phi$.

§**4.3 The Join.** In this section, we give a general algorithm that solves the global model checking problem for $\mathbf{T}_1 \otimes \mathbf{T}_2$. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be temporal logics and $\mathcal{M} = (W_1, \mathcal{R}_1, W_2, \mathcal{R}_2, V)$ be a model for $\mathbf{T}_1 \otimes \mathbf{T}_2$. We say that $\mathcal{M}$ is *finite* if W_1, W_2, $\mathcal{R}_1$ and $\mathcal{R}_2$ are finite, and, for every $(w_1, w_2) \in W_1 \times W_2$, $V((w_1, w_2))$ is finite. Let $\mathcal{M} = (W_1, \mathcal{R}_1, W_2, \mathcal{R}_2, V)$ be a *finite* $\mathbf{T}_1 \otimes \mathbf{T}_2$-model and

$\psi \in \mathcal{L}_{\mathbf{T}_1 \otimes \mathbf{T}_2}$. The *global model checking problem* for $\mathbf{T}_1 \otimes \mathbf{T}_2$ is to check whether there exist $w_1 \in W_1$ and $w_2 \in W_2$ such that $\mathcal{M}, w_1, w_2 \models_{\mathbf{T}_1 \otimes \mathbf{T}_2} \psi$.

Because of space limitations we have to omit the pseudo-code for a model checker for $\mathbf{T}_1 \otimes \mathbf{T}_2$ that exploits model checkers $\mathtt{MC}_{\mathbf{T}_1}$ and $\mathtt{MC}_{\mathbf{T}_2}$ for the component logics $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively; its code is similar to the code for the model checker $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$ given in Figure 4.2. For $\mathtt{MC}_{\mathbf{T}_1 \otimes \mathbf{T}_2}$ similar termination, soundness and completeness results may be obtained as for $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$.

5 Computational Complexity

We now turn to an analysis of the computational complexity of the model checkers proposed in the previous section. An instance for the model checking problem has two components: a model $(W, \mathcal{R}, V)$ and a formula ψ. In our analysis, we will consider three main complexity parameters: the cardinality n of W, the sum m of the cardinalities of the relations in $\mathcal{R}$, and the length k of ψ, i.e., the number of operators and proposition letters in ψ.

We will specify the complexity of the combined model checker in terms of that of the component model checkers. The complexity of the combined model checker is the sum of two factors: the communication overhead and the model checking cost. The *communication overhead* is the time spent for "packing" the inputs for the components and for "unpacking" their outputs; this represents the cost of the interaction between the components. The *model checking cost* represents the cost of performing the actual model checking of the component logics.

We first consider the case of temporalization. Let $\mathbf{L}$ be a logic and $\mathbf{T}$ a temporal logic. We write $C_{\mathbf{T}(\mathbf{L})}(\cdot,\cdot,\cdot)$ (resp. $C_{\mathbf{L}}(\cdot,\cdot)$, $C_{\mathbf{T}}(\cdot,\cdot,\cdot)$) for the complexity function of the model checker $\mathtt{MC}_{\mathbf{T}(\mathbf{L})}$ (resp. $\mathtt{MC}_{\mathbf{L}}$, $\mathtt{MC}_{\mathbf{T}}$). Note that $C_{\mathbf{L}}(\cdot,\cdot)$ has two parameters (the size of the model and the length of the formula).

Theorem 5.1 *Let $(W, \mathcal{R}, g)$ be a finite $\mathbf{T}(\mathbf{L})$-model and ψ a $\mathbf{T}(\mathbf{L})$-formula. The complexity of $\mathtt{MC}_{\mathbf{T}(\mathbf{L})}$ on input $\mathcal{M}$ and ψ is*

$$\mathcal{O}(n) \cdot [k \cdot C_{\mathbf{L}}(N, \mathcal{O}(1)) + C_{\mathbf{L}}(N, \mathcal{O}(k))] + C_{\mathbf{T}}(n, m, \mathcal{O}(k)),$$

where $n = |W|$, $m = \sum_{R \in \mathcal{R}} |R|$, $k = |\psi|$ and $N = \max_{w \in W} |g(w)|$.

The communication overhead is the cost of computing the set $\mathrm{MML}_{\mathbf{L}}(\psi)$ and the formula $\mathtt{abs}(\psi)$. It equals to $\mathcal{O}(k)$ and is dominated by the model checking cost. For instance, if $\mathbf{T}$ is CTL (hence $C_{\mathbf{T}}(n, m, k) = \mathcal{O}((n+m) \cdot k)$ [3]), and $\mathbf{L}$ is a logic such that $C_{\mathbf{L}}(n, k) = \mathcal{O}(n \cdot k)$, then the model checking cost is $\mathcal{O}(k \cdot (n \cdot N + m))$, hence still linear in the size of the model and in the length of the formula.

We now treat the independent combination of two temporal logics $\mathbf{T}_1$ and $\mathbf{T}_2$.

Theorem 5.2 *Let $\mathcal{M} = (W, \mathcal{R}_1, \mathcal{R}_2, V)$ be a finite $\mathbf{T}_1 \oplus \mathbf{T}_2$-model and ψ a $\mathbf{T}_1 \oplus \mathbf{T}_2$-formula. The complexity of $\mathtt{MC}_{\mathbf{T}_1 \oplus \mathbf{T}_2}$ on input $\mathcal{M}$ and ψ is:*

$$\begin{aligned}\mathcal{O}(m_1 + m_2 + n \cdot k) + \textstyle\sum_{i=1}^{2} \Big(& \mathcal{O}(k) \cdot C_{\mathbf{T}_i}(\mathcal{O}(n), \mathcal{O}(m_i), \mathcal{O}(1)) + \\ & \mathcal{O}(n) \cdot C_{\mathbf{T}_i}(\mathcal{O}(1), \mathcal{O}(1), \mathcal{O}(k)) + \mathcal{O}(1) \cdot C_{\mathbf{T}_i}(\mathcal{O}(n), \mathcal{O}(m_i), \mathcal{O}(k)) \Big),\end{aligned}$$

where $n = |W|$, $m_i = \sum_{R \in \mathcal{R}_i} |R|$, for $i = 1, 2$, and $k = |\psi|$.

The communication overhead is the cost of computing the connected components, of preparing the valuation as input to the model checking procedure, and of updating the valuations when the procedure returns. It adds up to $\mathcal{O}(m_1 + m_2 + n \cdot k)$, which is more significant than in the case of temporalization.

By way of example, if both $\mathbf{T}_1$ and $\mathbf{T}_2$ are CTL, and $m = m_1 = m_2$, then the communication overhead is $\mathcal{O}(m + n \cdot k)$, which is proportional to the model checking cost of $\mathcal{O}((n + m) \cdot k)$. So, the overall cost of the model checker for CTL $\oplus$ CTL is $\mathcal{O}((n + m) \cdot k)$, which is linear in the size of the model and the length of the formula.

Finally, we briefly consider the join of temporal logics $\mathbf{T}_1$ and $\mathbf{T}_2$. Due to space limitations we have to omit further discussions of Theorem 5.3 below.

Theorem 5.3 *Let $\mathcal{M} = (W_1, \mathcal{R}_1, W_2, \mathcal{R}_2, V)$ be a finite $\mathbf{T}_1 \otimes \mathbf{T}_2$-model and ψ a $\mathbf{T}_1 \otimes \mathbf{T}_2$-formula. Let $\overline{1} = 2$ and $\overline{2} = 1$. The complexity of* $\mathtt{MC}_{\mathbf{T_1} \otimes \mathbf{T_2}}$ *on input $\mathcal{M}$ and ψ is:*

$$\begin{aligned}&\mathcal{O}(n_1 \cdot m_2 + n_2 \cdot m_1 + n_1 \cdot n_2 \cdot k) + \\ &\quad \textstyle\sum_{i=1}^{2} \Big(\mathcal{O}(n_{\overline{i}}) \cdot [\mathcal{O}(k) \cdot C_{\mathbf{T}_i}(n_i, m_i, \mathcal{O}(1)) + C_{\mathbf{T}_i}(n_i, m_i, \mathcal{O}(k))]\Big),\end{aligned}$$

where $n_i = |W_i|$, $m_i = \sum_{R \in \mathcal{R}_i} |R|$, for $i = 1, 2$, and $k = |\psi|$.

6 Experimental Results

We briefly report on experimental results based on implementations of (combined) model checkers for $\mathrm{CTL}(\mathrm{CTL})$ and CTL $\oplus$ CTL. The model checkers have been implemented in C, and are available from `http://www.illc.uva.nl/~mdr/ACLG/Software/`. Tests were carried on a Sun ULTRA II (300MHz) with 1Gb RAM, under Solaris 5.2.5.

We tested our model checker for $\mathrm{CTL}(\mathrm{CTL})$ on 'linear' and 'dense' models. In our first test, we used $\mathbf{A}_1\mathbf{G}_1\mathbf{A}_2(p\mathbf{U}_2q)$ as a fixed test formula, and we varied the model $\mathcal{M}_1 = (W, R, g)$, where (W, R) is a complete binary tree of height h_1 and, for every $w \in W$, $g(w)$ is a labeled complete binary tree of height h_2. The outcomes are summarized in Table 6.1(a), where t_{ms} represents the CPU time in milliseconds. In the second test, we checked the formula $\mathbf{A}_1\mathbf{G}_1\mathbf{E}_2(p\mathbf{U}_2q)$ and used models $\mathcal{M}_2 = (W, R, g)$, where (W, R) is a complete graph of n_1 nodes and, for every $w \in W$, $g(w)$ is a complete graph of n_2 nodes. The outcomes are given in Table 6.1(b). The times listed in (b) are higher than those listed in (a) because $\mathcal{M}_2$ contains *dense* graphs, while $\mathcal{M}_1$ is based on *linear* graphs.

h_1	h_2	# nodes	# edges	t_{ms}
4	4	992	960	10
5	5	4032	3968	30
6	6	16256	16128	110
7	7	65280	65024	380
8	8	261632	261120	1490
9	9	1047552	1046528	5850

(a): trees and $\mathbf{A}_1\mathbf{G}_1\mathbf{A}_2(p\mathbf{U}_2q)$

n_1	n_2	# nodes	# edges	t_{ms}
32	32	1056	33792	20
64	64	4160	266240	110
128	128	16512	2113536	820
256	256	65792	16842752	6010
512	512	262656	134479872	47970
1024	1024	1049600	1074790400	386760

(b): complete graphs and $\mathbf{A}_1\mathbf{G}_1\mathbf{E}_2(p\mathbf{U}_2q)$

l	# nodes	# edges	t_{ms}
32	1024	1984	90
64	4096	8024	340
128	16384	32512	1400
256	65396	130560	5760
512	262144	532264	23480
1024	1048576	2095104	118980

(c): square grids

r	t_{ms}
0	1870
3	2540
7	2970
11	3860
15	4720
19	5590

(d): fixed square grid

Table 6.1: Experiments with combined model checkers.

Next, we treated the independent combination $\mathrm{CTL} \oplus \mathrm{CTL}$. Our test models were 'square grid' models, where we varied either the size of the model or the 'degree of interaction' of the formula. In the first test (Table 6.1(c)), we used $\mathbf{A}_1\mathbf{G}_1 q \wedge \mathbf{A}_2\mathbf{G}_2 q$ as our test formula on a square grid (W, R_1, R_2, V) of width l where rows are the connected components of (W, R_1) and columns the connected components of (W, R_2). For the second test (Table 6.1(d)), we used a square grid of width 256. The test formula was f_{20} (where $f_0 = q$, and $f_{k+1} = \mathbf{E}_i\mathbf{X}_i f_k$, for $k \geq 0$ and $i \in \{1, 2\}$), varying the number r of alternations of blocks of the form $\mathbf{E}_i\mathbf{X}_i\mathbf{E}_j\mathbf{X}_j$ $(i \neq j)$ occurring in f_{20} from 0 (no interaction) to 19 (maximal interaction). Increasing the degree of interaction in the formula increased the required computing time, thus confirming that the communication overhead is higher when checking formulas with a high degree of interaction.

7 An Application to Time Granularity

In this section we describe the relevance of our results to *temporal logics for time granularity*. Temporal logics have been successfully used for modeling and analyzing the behavior of (real-time) reactive systems. The behavior of reactive systems whose components have dynamic behaviors regulated by very different time constants, e.g., days, hours, and seconds, is naturally modeled by a set of differently-grained temporal domains. The addition of time granularity allows one to give concise specifications of such *granular reactive systems* (GRSs). The theory of finitely-layered structures for time granularity has been investigated in [14]. As for ω-layered structures, the theories of *upward (downward) unbounded layered structures,* i.e., ω-layered structures consisting of a finest (coarsest) temporal domain together with an infinite number of ever coarser (ever finer) domains, respectively, are non-elementarily decidable [13]. An expressively complete and elementarily decidable temporal logic counterpart of the theory of downward unbounded layered structures has been proposed in [7].

The behavior of a reactive system with respect to layered structures can be described as a suitable *combination* of temporal evolutions (sequences of states over a given temporal domain) and temporal refinements (mapping of a state, within a given domain, into a finite sequence of states over a finer temporal domain). As a consequence, both the model describing a system's operational behavior and the specification language can be obtained by *combining* simpler models and languages, and model checking procedures for combined logics can be used. For instance, a temporalized model $\mathcal{M} = (W, R, g)$ can be used in a 'horizontal' and a 'vertical' way for model checking purposes. Horizontally, it can be used to deal with finitely-layered, upward and downward unbounded GRSs. Assume that R is *linear,* that is, every node in W has at most one R-successor. The 'top' frame (W, R) models the granular relationships among the different components (wRv meaning that the component associated with v is a one-step refinement of the one associated with w), while, for $w \in W$, the model $g(w)$ captures the internal behavior of a single component. $\mathcal{L}_{\mathbf{T}_1(\mathbf{T}_2)}$ is the specification language, where $\mathbf{T}_1$ (resp. $\mathbf{T}_2$) is a *linear* (resp. *branching*) temporal logic. 'Horizontal' properties, that is, formulas that predicate over temporal evolutions, are easily expressible and verifiable in this framework, but it is hard to capture 'vertical' properties which refer to temporal refinements. In contrast, within the 'vertical' model checking framework one can deal with finitely-layered and downward unbounded GRSs. The interpretation of the model is different then: the 'top' frame (W, R) models the evolution of the coarsest component of the system, while, for $w \in W$, the model $g(w)$ captures the behavior of all temporal refinements of w. Specifications can be written in $\mathcal{L}_{\mathbf{T}_1(\mathbf{T}_2)}$, where $\mathbf{T}_1$ and $\mathbf{T}_2$ are branching temporal logics. As for expressiveness of the 'vertical' framework, the situation is dual to the 'horizontal' case.

8 Conclusions

We have addressed the problem of model checking for combined logics and structures. In contrast to combined deductive engines, combinations of model checking procedures are very well behaved, even in the presence of strong forms of interaction. In particular, complexity upper bounds transfer from the component framework to the combined one, and the introduced communication overhead is in most cases non significant with respect to the actual model checking cost.

One of the motivations for this work has been the need to develop model checking frameworks for granular reactive systems and logics. We have shown that this is indeed possible, using a *divide and conquer* strategy: we first isolated the orthogonal 'simple' entities in which a granular system can be decomposed. Then, we applied well-known structures and logics to the component entities. Although we only applied this *divide and conquer* approach to granular reactive systems, because of its generality, we feel that it can be useful to model and analyze many other complex systems, which, inherently, are the composition of simpler entities.

Acknowledgment. Massimo Franceschet and Angelo Montanari were supported by the MURST Project 'Saladin.' Maarten de Rijke was supported by the Spinoza project 'Logic in Action' and by a grant from the Netherlands Organization for Scientific Research (NWO), under project number 365-20-005.

References

[1] B. Bennett, C. Dixon, M. Fisher, E. Franconi, I. Horrocks, U. Hustadt, and M. de Rijke. Combining modal logics. *Submitted*, 1999.

[2] P. Blackburn and M. de Rijke. Editors' introduction. *Notre Dame Journal of Formal Logic*, 37:161–166, 1996.

[3] E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Trans. Programming Languages and Systems*, 8(2):244–263, 1986.

[4] K. Fine and R. Schurz. Transfer theorems for multimodal logics. In *Proceedings Arthur Prior Memorial Conference*, 1989.

[5] M. Finger and D.M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1:203–233, 1992.

[6] M. Finger and D.M. Gabbay. Combining temporal logic systems. *Notre Dame Journal of Formal Logic*, 37:204–232, 1996.

[7] M. Franceschet and A. Montanari. Branching within time. Technical Report UD/03/2000/RR, DIMI, Università di Udine, 2000.

[8] D. Gabbay and V. Shehtman. Products of modal logics, Part 1. *Logic Journal of the IGPL*, 6:73–146, 1998.

[9] D.M. Gabbay and M. de Rijke, editors. *Frontiers of Combining Systems 2*, volume 7 of *Studies in Logic and Computation*. Research Studies Press/Wiley, 2000.

[10] J.Y. Halpern and M.Y. Vardi. Model checking vs. theorem proving: a manifesto. In: *Proc. KR'91*, pages 325–334, 1991.

[11] E. Hemaspaandra. Complexity transfer for modal logic. In *Proc. LICS'94*, pages 164–173, 1994.

[12] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *Journal of Symbolic Logic*, 56:1469–1485, 1991.

[13] A. Montanari, A. Peron, and A. Policriti. Decidable theories of ω-layered metric temporal structures. *Logic Journal of the IGPL*, 7(1):79–102, 1999.

[14] A. Montanari and A. Policriti. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic*, 37:260–282, 1996.