To appear in Formal Aspects of Computing

Requirements, specifications and minimal refinement

Nikos Gorogiannis[†] and Mark Ryan[‡]

[†] School of Computing, University of the West of England
[‡] School of Computer Science, University of Birmingham

1. Introduction

The area of automatic verification has, in the recent past, provided us with feasible methods for automatically checking whether a design, representing a system, satisfies a set of requirements (see e.g. [CE81, BCM+90, McM93]). However, the *development* of system designs (usually expressed as transition systems) is not trivial. The usual cycle of design–debug–re-design is time-consuming and error prone, partly due to the fact that the incorporation of a requirement in an existing design is a non-trivial, largely unformalised process.

Within the context of design development, it is often the case that a design that satisfies a set of requirements already exists and a new requirement is to be 'added', in some sense, to that design. This may happen either when a new system is being designed with an existing one as a basis, or while incrementally designing a system. In both cases, it is desirable to have a methodology for automatically obtaining a design out of the old one and the new requirement. Moreover, this process should be theoretically well-understood in terms of guarantees of how exactly the new design relates to the original one and the requirement to be added.

There are many types of such changes, corresponding to the different meanings assigned to the process of 'adding' a requirement to a design of a system. Therefore, one cannot possibly hope to create a meaningful, unified theoretical framework that encompasses any and all types of design change. In this paper, we will focus on a specific type that we believe is useful in the process of designing systems, which we call minimal refinement.

Example 1. An access control system for a database application is being designed. To aid in its verification, an abstraction has been constructed in the form of a state transition system. Each possible computation path exhibited by the transition system corresponds to an allowable transaction with the database. At some point, a further rule is imposed so that, e.g., a certain kind of transaction is necessarily followed by another. This rule is to be applied to the original system as an add-on and, as such, it is preferable not to re-design the abstraction from scratch, but re-use the existing model as a starting point. Since the new system circumscribes the allowable behaviours of the original system, what we are looking for is a refinement of that model that satisfies the new rule.

Example 2. In an alternative scenario, imagine that a model of a system has been constructed and verified with a model checker. It is now desirable to produce a version of this model which is more low-level so that it can be used for the automatic construction of code that implements this system. The new, more 'concrete'

Correspondence and offprint requests to: Mark Ryan, School of Computer Science, University of Birmingham, B15 2TT, UK. e-mail: M.D.Ryan@bham.ac.uk.

version of the system may contain implementation-level details and states. Again, we are interested in producing this refined version automatically, while preserving the behaviours of the specification that are admissible by the implementation mechanisms present in the result.

In both scenarios we seek to produce a *refinement* of a system. Refinement (see e.g. [AL91]) can be viewed as behaviour-containment; if A and B are two systems, then

A refines B iff behaviours(A) $\subseteq$ behaviours(B).

We will be using transition systems to represent designs of systems (and from now on we will use the terms system and model interchangeably), and as such, an appropriate notion of behaviour is the computation tree. Correspondingly, the set of behaviours exhibited by a transition system is the set of computation trees that results from the unravelling of the transition system. More specific definitions will be given in the following section.

In both examples, a refinement of a design is sought. However, refining a system yields an implementation, or a more 'concrete' version of it, but not necessarily a useful one: depending on the formalism used, it is frequently the case that trivial models exist that refine almost every other model. For example, in the case of the database access control system, it is true that an empty transition system does not allow any behaviours at all and, therefore, refines the original one; it is hardly a useful one though. In other words, we are interested in refining the original model but only so much as is necessary in order to satisfy a given guiding property. In this way, behaviours of the original system are only sacrificed if necessary.

Thus, instead of just looking for designs that refine the initial one and satisfy the new requirement, we will use refinement to *order* the designs. We are led, then, to the concept of *minimal refinement*. Minimal refinement can be used whenever the designer has a model of a system that already circumscribes the allowed behaviours of the system. Then, a new property can be applied and a new design obtained, that satisfies the new requirement, refines the initial design and exhibits as many of its behaviours as possible. A corollary of this is that by using minimal refinement we get automatic preservation of safety properties.

The contributions of this paper lie, firstly, in the introduction of the notion of minimal refinement as a method for the stepwise addition of requirements in a model. Secondly, we investigate and prove results concerning several issues around minimal refinement such as the soundness and decidability of algorithms for computing minimal refinements.

In what follows we will define this process and study its theoretical and applied aspects. We will focus on representations of designs based on transition systems, and on modal and temporal logics as languages for expressing requirements. We begin by introducing the theoretical background in Section 2. Then, we will proceed to formalise these intuitions and define minimal refinement in a precise way in section 3. In the same section, the problems that arise from our definition are discussed, and specific technical questions aiming at resolving those problems are specified. These questions are investigated in the context of modal logic in Section 4. In the aim of addressing expressiveness issues related to modal logic, we further develop the investigation of these questions into the realm of temporal logic in Section 5. Finally, we conclude and summarise the possible avenues for extending the work presented in this paper, in Section 6. The definitions behind minimal refinement and part of the results presented in Section 4 have appeared in [GR02]. The results presented in Section 5 have appeared in [Gor03].

2. Preliminaries

2.1. General background

Before discussing specific logics, we will approach the subject from a higher level. Therefore, let $\mathcal{L}$ and $\mathcal{M}$ be sets, corresponding to the set of formulae and the set of models of a logic. Let, also, $\models \subseteq \mathcal{M} \times \mathcal{L}$ be a satisfaction relation. The class of models that satisfies a formula ϕ will be denoted by $\mathrm{mod}(\phi)$. Let $\leq \subseteq \mathcal{M} \times \mathcal{M}$ be a preorder (i.e., a transitive and reflexive relation) on models. The strict counterpart of $\leq$, denoted by $<$ is defined by the condition $M < N$ iff $M \leq N$ and $N \not\leq M$. Given a set $S \subseteq \mathcal{M}$, the minimisation operator is defined in the usual way, $\min_{\leq}(S) = \{M \in S \mid \forall N \in S, N \not< M\}$. As noted in the introduction, we will be looking at an operation of the form of $\min_{\leq}(\mathrm{mod}(\phi))$.

This operation resembles one often found in the areas known as theory (or belief) change (see, e.g., [Gro88, Dal88, KM89, KM91]) and non-monotonic reasoning (e.g, [KLM90, BMP97, BEF93]), among others.

The shared intuition is that, when given a property ϕ in some logic, instead of selecting all the structures that satisfy ϕ ($\mathrm{mod}(\phi)$) we employ some extra-logical information and treat some models inside $\mathrm{mod}(\phi)$ in a *preferential* way. This information takes frequently the form of a relation over models and is usually called a *preference relation*.

2.2. Modal logic

In Section 4, we will generally work with a finite set $\mathcal{A}$ of propositional variables. The modal language $\mathcal{L}_{\mathrm{K}}$ of the logic K_m on $\mathcal{A}$ with m modalities is defined inductively; if $p \in \mathcal{A}$ then $p \in \mathcal{L}_{\mathrm{K}}$; if ϕ and ψ are in $\mathcal{L}_{\mathrm{K}}$ then so are $\neg\phi$ and $\phi \wedge \psi$; if $\phi \in \mathcal{L}_{\mathrm{K}}$ then $\Diamond_i \phi \in \mathcal{L}_{\mathrm{K}}$ for all $1 \leq i \leq m$. The usual propositional abbreviations apply as well as the modal $\Box_i \equiv \neg\Diamond_i\neg$. The axiomatisation of K_m follows.

P. Any propositional tautology is an axiom.
K. Any formula of the form $\Box_i(\phi \to \psi) \to (\Box_i\phi \to \Box_i\psi)$ with $1 \leq i \leq m$ is an axiom.

Its rules of inference are:

MP. Modus ponens: if ϕ and $\phi \to \psi$, then ψ.
Nec. Necessitation: if ϕ, then $\Box_i\phi$, for all $1 \leq i \leq m$.

The *logic* Λ of K_m is defined to be the smallest subset of $\mathcal{L}_{\mathrm{K}}$ that contains all the instances of the above axioms and is closed under the two rules of inference. The fact that a formula $\phi \in \mathcal{L}_{\mathrm{K}}$ is in Λ is denoted by $\vdash \phi$. If T is a set of sentences and ϕ a sentence, then ϕ is *deducible from* T, written $T \vdash \phi$, if there exists a number $n \geq 0$ and sentences $\psi_1, \ldots \psi_n \in T$ such that $\vdash \psi_1 \wedge \ldots \wedge \psi_n \to \phi$. The set T is called *consistent* if $T \nvdash \bot$ and *inconsistent* otherwise.

The most popular semantics for modal logics is through *Kripke models*.

Definition 1. A tuple $M = \langle W_M, r_M, R^1_M, \ldots, R^m_M, v_M \rangle$ is called a Kripke model whenever

- W_M is a set of *states* or *worlds*.
- r_M is a distinguished state in W_M called the *initial state* or the *root*.
- $R^i_M \subseteq W_M \times W_M$ are *accessibility relations*. We will only consider models whose states are reachable from the root. In other words, for any state $s \in W_M$ there exists a sequence of states $s_1, \ldots, s_n$ such that $s_1 = r_M$, $s_n = s$ and for all $1 \leq j < n$, $(s_j, s_{j+1}) \in \bigcup_{i=1}^m R^i_M$.
- $v_M : W_M \to 2^{\mathcal{A}}$ is a *valuation* for the propositional letters.

By $|M|$ we denote the cardinality of W_M. The model M is said to be finite if $|M|$ is finite. The class $\mathcal{K}$ is the class of all Kripke models.

Satisfaction of formulae at a state s is defined inductively by the usual propositional clauses along with the modal one: $M, s \models \Diamond_i\phi$ iff $\exists t \in W_M$ such that $(s,t) \in R^i_M$ and $M, t \models \phi$. We will write $s \models \phi$ when the model is obvious. Satisfaction at the level of models is defined as follows.

- $M \models \phi$ iff $r_M \models \phi$,
- $M \models_G \phi$ iff $\forall s \in W_M, s \models \phi$ (*global satisfaction*).

We will write $M \models T$ for a set of sentences T whenever M satisfies all formulae in T. Semantic entailment is defined as follows: if ϕ is a sentence and T a set of sentences, T *semantically entails* ϕ, written $T \models \phi$ iff for all models $M \in \mathcal{K}$, if $M \models T$ then $M \models \phi$. The class of models that satisfies a formula ϕ is denoted by $\mathrm{mod}_{\mathcal{K}}(\phi)$ and the class of models that globally satisfy a formula ϕ, by $\mathrm{mod}^G_{\mathcal{K}}(\phi)$ (similarly for sets of sentences). A set of sentences T is called *satisfiable* iff $\mathrm{mod}_{\mathcal{K}}(T) \neq \emptyset$ (similarly for *globally satisfiable*). The set of sentences true at a state s is denoted by $\mathrm{th}(s)$. The theory of a model is defined as the theory of its root, $\mathrm{th}(M) = \mathrm{th}(r_M)$. Two models M, N are *logically equivalent* iff $\mathrm{th}(M) = \mathrm{th}(N)$.

A few well-known facts about K_m and its logic are summarised below [BdRV01, Che80]. The set T stands for a set of sentences and ϕ for a sentence of $\mathcal{L}_{\mathrm{K}}$.

- $T \models \phi$ iff $T \vdash \phi$ (*soundness* and *strong completeness* of Λ with respect to $\mathcal{K}$).
- T is satisfiable iff T is consistent.
- If $T \vdash \phi$ and $\phi \vdash \psi$ then $T \vdash \psi$.

We will now turn to the semantical notions of *bisimulation* and *simulation*.

Definition 2. Let M, N be models and $B \subseteq W_M \times W_N$ a non-empty relation. The relation B is a *bisimulation* if

- it relates the initial states, $(r_M, r_N) \in B$,
- it respects the valuations, $(s,t) \in B$ implies $v_M(s) = v_N(t)$,
- if $(s,t) \in B$ and s' is an R^i_M-successor of s then there exists t', an R^i_N-successor of t, such that $(s',t') \in B$, for all $1 \leq i \leq m$ (the *forth* condition),
- if $(s,t) \in B$ and t' is an R^i_N-successor of t then there exists s', an R^i_M-successor of s, such that $(s',t') \in B$, for all $1 \leq i \leq m$ (the *back* condition).

If there exists a bisimulation between M, N then M and N are called *bisimilar*, written $M \sim N$ and it follows that $\mathrm{th}(M) = \mathrm{th}(N)$. The converse is not generally true, but it does hold for the classes of m-saturated and finite models which we will concerns ourselves with later.

Definition 3. Let M, N be models and $S \subseteq W_M \times W_N$ a non-empty relation on their state-spaces. The relation S will be called a *simulation* iff it satisfies the first three clauses in the definition of bisimulation, i.e. it must link the initial states, preserve valuations and respect the accessibility relations but in one-way only (the forth condition).

If there exists a simulation from M to N we write $M \to N$ or $N \leftarrow M$ and say that N simulates M or that M is simulated by N. Whenever $M \leftarrow N$ and $M \to N$ we will say that M and N are *similar* or *simulation equivalent* and write $M \leftrightarrows N$. It is easy to check that simulations are transitive and reflexive. If two models are bisimilar then it follows that they are simulation-equivalent as well, although the converse does not hold in general.

A formula is called *positive universal* iff it is made up only from propositional letters, their negations, the propositional connectives $\wedge$ and $\vee$ and the universal modality $\Box_i$. $\mathcal{L}_{\mathrm{PU}}$ is the subset of $\mathcal{L}_{\mathrm{K}}$ that consists of positive universal formulae. If s is a state then $\mathrm{PU}(s) = \mathcal{L}_{\mathrm{PU}} \cap \mathrm{th}(s)$. If M is a model, then $\mathrm{PU}(M) = \mathrm{PU}(r_M)$. Dually, a *positive existential* formula is made up from $p, \neg p, \wedge, \vee$ and $\Diamond_i$. $\mathcal{L}_{\mathrm{PE}}$ and PE are defined similarly and are duals of $\mathcal{L}_{\mathrm{PU}}$ and PU respectively. Note that the negation of a PU formula is a PE one and vice versa. If P is a set of PU sentences then P^c is the complement of P with respect to $\mathcal{L}_{\mathrm{PU}}$, i.e. $P^c = \mathcal{L}_{\mathrm{PU}} \setminus P$. The set $\overline{P}$ is defined as the set that contains the negation of every formula in P, i.e. $\overline{P} = \{\neg\phi \mid \phi \in P\}$.

We define $\Box^*\phi$ to denote a set of sentences as follows,

$$\Box^*\phi = \{\Box_{i_1} \ldots \Box_{i_n} \phi \mid \forall n, j, i_j (n \geq 0 \wedge 1 \leq j \leq n \wedge 1 \leq i_j \leq m)\}.$$

The set $\Box^*T$ is defined similarly, but on sets of sentences rather than on formulae:

$$\Box^*T = \bigcup_{\phi \in T} \Box^*\phi.$$

If a model satisfies $\Box^*T$ at its starting state then, obviously, it will have to satisfy T on all the states reachable from the root. Therefore, because of the condition that we have imposed on the reachability of all states in a model, it follows that $\mathrm{mod}^G_{\mathcal{K}}(T) = \mathrm{mod}_{\mathcal{K}}(\Box^*T)$.

Apart from the class of finite Kripke models, one other class will be of particular interest to us, the class of m-saturated models. Essentially, an m-saturated model is a Kripke model that satisfies a condition similar to *compactness* across the successors of its states. The formal definition follows.

Definition 4. Let M be a model, s a state in W_M and T a set of sentences.

1. T will be called *satisfiable on the successors of* s iff for each relation R^i_M there exists a state $t \in W_M$ such that $(s,t) \in R^i_M$ and $T \subseteq \mathrm{th}(t)$.
2. Similarly, T will be called *finitely-satisfiable on the successors of* s iff for each relation R^i_M and for any finite set of sentences $F \subseteq T$ there exists an R^i_M-successor t of s such that $F \subseteq \mathrm{th}(t)$.
3. A state s is called *m-saturated* iff, for any set of sentences T, if T is finitely-satisfiable on the successors of s, then it is satisfiable on the successors of s.
4. A model is m-saturated if all its states are m-saturated.

MSAT will denote the class of m-saturated models. Notice that MSAT is closed under bisimulation.

The following two lemmas characterise simulation in syntactic terms, and establish an exact match in the m-saturated case. The first one states that simulation implies PU-language inclusion.

Lemma 1. If M, N are models such that $M \leftarrow N$, then $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$ (Folklore).

The second lemma concerns the effect of PU-language inclusion, in the case of m-saturated models.

Lemma 2. Let M, N be models. If $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$ and M is m-saturated, then there exists a simulation from N to M, $M \leftarrow N$ (Folklore).

Proof. For convenience we will work with PE formulae, the dual of positive universal ones. Note that $\mathrm{PU}(s) \subseteq \mathrm{PU}(t)$ iff $\mathrm{PE}(s) \supseteq \mathrm{PE}(t)$. Define a relation S such that

$$(s,t) \in S \quad \text{iff} \quad s \in W_N, t \in W_M \text{ and } \mathrm{PE}(s) \subseteq \mathrm{PE}(t).$$

We prove that S is a simulation.

- Since $\mathrm{PE}(r_N) = \mathrm{PE}(N)$, $\mathrm{PE}(N) \subseteq \mathrm{PE}(M)$ and $\mathrm{PE}(M) = \mathrm{PE}(r_M)$, we have $(r_N, r_M) \in S$.
- Since $\mathcal{A} \cup \overline{\mathcal{A}} \subseteq \mathcal{L}_{\mathrm{PE}}$, S respects the valuations, i.e. if $(s,t) \in S$ then $v_N(s) = v_M(t)$.
- Assume that state s has a successor s' with respect to a relation R^i_N. Let P be the set of PE sentences of s'. For any finite subset $F \subseteq P$, $s' \models \bigwedge F$ and thus $s \models \Diamond_i \bigwedge F$. The formula $\Diamond_i \bigwedge F$ is a PE formula, so by definition it is satisfied at t. Thus there is an R^i_M-successor of t that satisfies $\bigwedge F$. In other words, P is finitely-satisfiable on the successors of t. The model M however is m-saturated, thus there is an R^i_M-successor t' of t that satisfies P and as such $\mathrm{PE}(s') \subseteq \mathrm{PE}(t')$.

□

2.3. The logic ACTL

The logic ACTL [GL94] is a fragment of CTL [CES86], a branching-time temporal logic well-known for its use in model checking. As in the case of modal logic, $\mathcal{A}$ will be a finite set of atomic propositions.

Definition 5. The language $\mathcal{L}_{\mathrm{ACTL}}$ is defined inductively: for all $p \in \mathcal{A}$, $p \in \mathcal{L}_{\mathrm{ACTL}}$ and $\neg p \in \mathcal{L}_{\mathrm{ACTL}}$; if $\phi, \psi \in \mathcal{L}_{\mathrm{ACTL}}$ then $\phi \wedge \psi \in \mathcal{L}_{\mathrm{ACTL}}$ and $\phi \vee \psi \in \mathcal{L}_{\mathrm{ACTL}}$; finally, if $\phi, \psi \in \mathcal{L}_{\mathrm{ACTL}}$ then the formulae

- $\mathrm{AX}\phi$ (*on all successors,* ϕ),
- $\mathrm{A}(\phi \mathrm{U} \psi)$ (*on all paths,* ϕ *until* ψ),
- $\mathrm{A}(\phi \mathrm{R} \psi)$ (*on all paths,* ϕ *release* ψ),

are all formulae in $\mathcal{L}_{\mathrm{ACTL}}$.

The following abbreviations apply:

- $\mathrm{AG}\phi \equiv \mathrm{A}(\bot \mathrm{R} \phi)$ (*globally* ϕ) and
- $\mathrm{AF}\phi \equiv \mathrm{A}(\top \mathrm{U} \phi)$ (*eventually* ϕ).

Models for ACTL are structures similar to Kripke models.

Definition 6. A tuple $M = \langle W_M, S_M, \mathcal{A}_M, v_M, \rightarrow_M, \mathcal{F}_M \rangle$ is a model for ACTL with fairness constraints whenever

- W_M is a finite set of states,
- $S_M \subseteq W_M$ is a set of initial states,
- $\mathcal{A}_M \subseteq \mathcal{A}$ is a set of atomic propositions,
- $v_M : W_M \rightarrow 2^{\mathcal{A}_M}$ is a valuation,
- $\rightarrow_M \subseteq W_M \times W_M$ is the accessibility (or transition) relation,
- $\mathcal{F} \subseteq 2^{W_M}$ is a set of fairness conditions.

It is obvious by the above definition that only finite models will be considered. Notice, also, that there is a set of initial states instead of a single one. Another point of departure from the modal case is the incorporation of the atomic propositions into the model. Notice also that we restrict the number of accessibility relations to one: this relation can be viewed as the union of all the action-representing accessibility relations. As is customary in the temporal logic literature we will ignore these and focus on their union. Moreover, there is no necessity for the states to be reachable by the initial states through the transition relation: since the notion of satisfaction (defined below) is only local to the initial states, there is no danger of 'referring' to unreachable states.

As previously, $|M|$ will denote the cardinality of the state-space of M. The class of all such models will be denoted by $\mathcal{M}_{\text{FTSF}}$, with FTSF standing for finite transition systems with fairness constrains. An infinite sequence of states $\pi = s_0 s_1 \dots$ is a *path* in a model M iff for all $i \geq 0$, $s_i, s_{i+1} \in W_M$ and $s_i \rightarrow_M s_{i+1}$. The i-th state in π is denoted by π^i. For a path π in M, $\inf(\pi) \subseteq W_M$ is the set of states π visits an infinite number of times. A path π in M is *fair* iff for all $P \in \mathcal{F}_M$, $\inf(\pi) \cap P \neq \emptyset$. Essentially, a path is fair if it visits all the fairness condition sets infinitely often.

Satisfaction of ACTL formulae is defined as follows.

Definition 7. Let M be a model and $s \in W_M$ a state. Satisfaction of an ACTL formula on s is defined inductively:

- The usual definitions apply for satisfaction of an atomic proposition, a negated atomic proposition, conjunctions and disjunctions.
- $M, s \models \text{AX}\phi$ if and only if, for all fair paths π in M that start at s, $M, \pi^1 \models \phi$. The prefix AX corresponds to the modal $\Box$ with the difference that only the successors along fair paths are considered.
- $M, s \models \text{A}(\phi \text{U} \psi)$ if and only if, for every fair path π in M that starts at s, there is an i such that $M, \pi^i \models \psi$ and for all $j < i$, $M, \pi^j \models \phi$. In other words, ϕ *until* ψ.
- $M, s \models \text{A}(\phi \text{R} \psi)$ if and only if for every fair path π in M that starts at s, for all i, $M, \pi^j \not\models \phi$ for all $j < i$, implies $M, \pi^i \models \psi$. This is the dual of *until* in the sense that in full CTL it is the case that $\text{A}(\phi \text{R} \psi) \leftrightarrow \neg \text{A}(\neg\phi \text{U} \neg\psi)$.

We will write $M \models \phi$ iff for all $s \in S_M$, $M, s \models \phi$ (whether the modal or temporal satisfaction relation is meant by $\models$ will be made clear by the context). Accordingly, $\text{mod}_{\text{FTSF}}(\phi) = \{M \in \mathcal{M}_{\text{FTSF}} \mid M \models \phi\}$. Similarly with modal logic, for any two formulae of ACTL, $\phi \models \psi$ means that for any model $M \in \mathcal{M}_{\text{FTSF}}$, $M \models \phi$ implies $M \models \psi$.

Next, the definition of fair simulation is presented. This concept was introduced in [GL94] where it was called a homomorphism and is also known as $\exists$-simulation [HKR97].

Definition 8. Let A, B be models with $\mathcal{A}_A \supseteq \mathcal{A}_B$ and α, β be states in A, B respectively. A relation $H \subseteq W_A \times W_B$ is a fair simulation from (A, α) to (B, β) iff

1. $(\alpha, \beta) \in H$,
2. For all $a \in W_A, b \in W_B$, $(a, b) \in H$ implies
   - $v_A(a) \cap \mathcal{A}_B = v_B(b)$,
   - For every fair path $\pi = a_0 a_1 \dots$ in A with $a_0 = a$ there exists a fair path $\pi' = b_0 b_1 \dots$ in B with $b_0 = b$ such that for every i, $H(a_i, b_i)$.

H is a fair simulation from A to B, denoted $B \leftarrow A$, iff for all $\alpha \in S_A$ there is a $\beta \in S_B$ such that $(\alpha, \beta) \in H$.

There are several differences with simulation, as seen in the previous section. Firstly, there may be initial states in B that do not correspond to (initial) states in A, something not possible with simulation. Secondly, there may be successors of a state $a \in W_A$ that do not belong to any fair path beginning at a. In that case, these successors do not impose any restrictions in the refinements of A as they necessarily would in the case of simulation.

ACTL consists purely of positive universal formulae. Therefore, fair simulation trivially implies language-inclusion, i.e. $A \leftarrow B$ implies $\text{th}(A) \subseteq \text{th}(B)$. The converse is true as well, since we are only considering finite models.

3. Minimal refinement of models of systems

In this section, we will make precise the notion of minimal refinement. Its exposition in the examples given earlier is under-specified in that concepts such as behaviour, model and requirement have not been formally defined. For the purposes of introduction, general definitions will be used to present the idea while specific instances of this framework are formally developed fully in Sections 4 and 5.2.

Let $\leftarrow \subseteq \mathcal{M} \times \mathcal{M}$ be a preorder on models that represents *refinement*. We will write $M \leftarrow N$ when N refines M, and $M \leftrightarrows N$ when M, N are refinement-equivalent. Then, the following ordering may be defined.

Definition 9. Let M, A, B be models. Then, $A \leq_M B$ iff

1. $M \leftarrow A \leftarrow B$ or
2. $M \leftarrow A$ but $M \not\leftarrow B$ or
3. $A \leftrightarrows B$.

The aim of this ordering is to provide a measure of how 'concrete' is a refinement of a model: if $A <_M B$ then A refines M and either B refines A (while the converse is not true), or B does not refine M. In the first case A is a less 'concrete' refinement of M than B is. The same applies in the second case but in a trivial sense.

One way of seeing this definition which might be appealing to some readers is to consider, for some M, the set of models $\mathcal{M}_M = \{N \mid M \leftarrow N\}$. Then we may write

$$\leq_M = \leftarrow|_{\mathcal{M}_M} \cup (\mathcal{M}_M \times \overline{\mathcal{M}_M}) \cup \leftrightarrows|_{\overline{\mathcal{M}_M}}$$

Here, overline denotes set complementation, and the vertical bar denotes restricting a relation to a subset of the domain/range. As in the definition, $\leq_M$ is seen to consist of three components. The first component indicates that $\leq_M$ coincides with refinement in $\mathcal{M}_M$ (condition 1). If $M \not\leftarrow N$ then N becomes a maximal element in $\leq_M$ (condition 2). Finally, on these maximal elements we only allow simulation equivalence (part of condition 3, and equivalent to it in the context of condition 1).

It is not hard to see that this ordering is reflexive and transitive. As such, it is a preorder. Using $\leq_M$, minimal refinement may now be defined in a formal way.

Definition 10. Let M be a model in $\mathcal{M}$ and $\phi \in \mathcal{L}$ a formula. We define the minimal refinement operation $*$ as

$$M * \phi = \min_{\leq_M}(\mathrm{mod}(\phi)),$$

i.e. the set of models in $\mathrm{mod}(\phi)$ which are $\leq_M$ minimal. Note that $\leq_M$ is a preorder and that the minimisation works in an identical fashion to minimising over a partial order.

This definition resembles a particular type of theory change known as update [KM92]: it is a point-wise definition, i.e. the ordering is indexed by a model, in contrast with the case of belief revision, where the index is an arbitrary theory [KM91].

Given such an operation, several questions arise. Generally, the existence of minimal elements within a set with respect to an ordering is not guaranteed and, as such, it is not obvious that a minimal refinement will yield any models. Therefore, before studying further a particular instantiation of this framework, the conditions that guarantee the existence of minimal models must be investigated. In order to address this issue, we employ a version of the notion of *stoppered*ness, a concept known from the non-monotonic reasoning literature, which is related to, but weaker than that of *well-foundedness*.

Definition 11. An ordering $\leq$ over $\mathcal{M}$ is *stoppered* over a collection of sets of models $\mathcal{C} \subseteq 2^{\mathcal{M}}$ iff for each $X \in \mathcal{C}$ and any model $A \in X$ there exists a model $B \in X$ such that $B \leq A$ and B is $\leq$-minimal in X.

Not all pairs of model M and formula ϕ will yield interesting minimal refinements. For example, assume that there is no model of ϕ that refines M. In that case $\leq_M$ will be flat inside $\mathrm{mod}(\phi)$ in the sense that for any two models $A, B \in \mathrm{mod}(\phi)$, $A \leq_M B$ iff $A \leftrightarrows B$. As a result, $M * \phi = \mathrm{mod}(\phi)$. Obviously, finding a general condition that will exclude degenerate cases such as this is very difficult given that crucial notions such as the logic and the refinement relation are undefined at this stage. A significant category of such cases can be captured by the general condition presented above, that is, when there is no model of ϕ that refines M. We call this case trivial and its negation, non-trivial:

Definition 12. Let M be a model in $\mathcal{M}$ and $\phi \in \mathcal{L}$ a formula. We call $M * \phi$ *non-trivial* iff there exists a model N such that $M \leftarrow N$ and $N \models \phi$.

Thus, it is of interest to know the conditions that guarantee non-triviality of the results of the operation.

Suppose that we have designed a model M that we want to minimally refine with a new property ϕ. Since it is not guaranteed that a model of ϕ that refines M exists, before applying minimal refinement we would like to know whether $M * \phi$ is trivial or not. Is there an algorithm that can compute this and under which conditions?

Having secured the non-triviality of $M * \phi$ and assuming stopperedness holds, the goal, of course, is to *compute* the models that minimally refine M under ϕ. Intrinsic to this problem is the existence of an algorithm that, given two models M and N and a formula ϕ, decides the minimality of N with respect to M and ϕ, i.e. whether $N \in M * \phi$ or not.

Finally, in some situations it is desirable to be able to answer conditional queries about the minimal refinement. For example, we may want to know whether a given model M, when minimally refined by ϕ, will entail a certain property ψ. This amounts to the existence of an algorithm for answering queries of the form $M * \phi \models \psi$.

4. Minimal refinement in modal logic

4.1. Introduction

Imagine that for the purposes of model checking a microprocessor, a transition system has been constructed that models, for example, the interface between the microprocessor (CPU) and the memory management unit (MMU). One way to distinguish between the state transitions of the different sub-systems is to have many transition relations in the model, one for each sub-system. Suppose further that an error has been discovered in the model, such that a particular CPU transition is followed by an insufficient number of 'wait-states' before the MMU has reached an appropriate state of its own. Also, the model is considered to be sufficient in other cases. It would be desirable to refine this model into another which does not contain this problematic scenario, possibly by unfolding existing state-loops, while avoiding the addition of 'new' transition sequences. The framework presented in Section 3 could, then, be used for the automatic generation of such a refinement.

In another scenario, suppose that in order to study mathematically (and possibly model check) a board game of two players, we wish to construct a transition system that models all or some of the possible evolutions of the game. This transition system would have two transition relations, a black and a white one, each one corresponding to a player. Suppose that we are interested in studying the effect of adding a new rule to the game, of the form "if the white player has captured the centre of the board, then the black player must retreat by one square". Since new game-state sequences should not be added by the incorporation of this rule, what we are really looking for is a refinement of the model we started out with. Moreover, this refinement should be minimal, as the addition of this rule should only exclude evolutions of the game that are expressly forbidden.

In both of these cases, modal logic can be used to phrase the requirements presented. For example, in the second one, a formula of the form $\mathsf{CentreCapturedBy}(\mathit{white}) \rightarrow \Box_{\mathit{black}}\mathsf{Retreats}(\mathit{black})$ is sufficient. More generally, modal logic can be used as a simple specification language for processes. It provides the mechanisms to describe what the state of a process is, what actions are possible and what actions are necessary from that state. Indeed, modal logic can be seen as Hennessy-Milner logic [HM85] extended with propositional operators and state-information.

In the universe of transition systems and modal logic, a behaviour of a model is a *computation tree*. If refinement is seen as containment of behaviours, then the appropriate concept in this case is *simulation* [Mil71, BFG+91]. This is the notion of refinement with which we will instantiate the ordering $\leq_M$ and minimal refinement in this section.

Our main interest is in the applications of minimal refinement; therefore, and especially in the context of model checking, the class of finite models is central to this work due to the fact that finite-state processes can be described in a straightforward way as finite models. However, a detour will be taken through another class of models, before the discussion of the finite case. This class, the modally-saturated models (referred to as m-saturated from now on) is not a class that directly lends itself to practical uses of modal logic, since

it includes infinite and uncountable models, and only imposes modest restrictions on its members. On the other hand, since it combines productively with the notion of the ultrafilter construction (more on which will follow), it will provide us with a set of results that will serve as tools for approaching the finite case. Moreover, it possesses the following properties [Hol95]:

- It subsumes the class of image-finite models (and hence the finite ones).
- It has the Hennessy-Milner property. (A class of models has the *Hennessy-Milner property* whenever for every pair of its models, they are bisimilar if and only if they are logically equivalent.)
- It is maximal in the sense that no proper superclass of it has the Hennessy-Milner property.
- It has also been used to provide semantics for simple process algebras.

The outline of this section is as follows. Firstly, the definition of minimal refinement is examined in the modal case, and examples motivate the related questions, in Section 4.2. Then, minimal refinement is examined in the case of m-saturated models in Section 4.3, where the questions of non-triviality and stopperedness are addressed. The case of minimal refinement of finite models is investigated in Section 4.4, where the results on stopperedness, non-triviality and decidability of several problems are discussed. Finally, related work is discussed in Section 4.5.

4.2. Minimal refinement in the modal case

In modal logic, we will define refinement as simulation (see Definition 3). As before, we define $\leq_M$ to measure the degree of refinement of M (Definition 9), as follows. Let M, A and B be models. Then, $A \leq_M B$ if

1. $M \leftarrow A \leftarrow B$ or
2. $M \leftarrow A$ but $M \not\leftarrow B$ or
3. $A \leftrightarrows B$,

where $\leftarrow$ means simulation.

Notice that the definition of minimal refinement,

$$M * \phi = \min_{\leq_M}(\mathrm{mod}(\phi))$$

is essentially parameterised over two dimensions here. Firstly, there are two meanings for the mod operator, the local and the global, as defined in Section 2.2. Secondly, since we are going to be concerned with several classes of models, the mod operation will range over the appropriate class. We will add some notation to make these issues clear. The $*$ symbol and the mod operator will be subscripted with the class of models in question. In addition, the superscript G when it is present will indicate that only the models that satisfy globally the given property are to be considered. Finally, the notation $[G]$ will indicate that the result under discussion applies to both the local and the global case. So, for example, $\mathrm{mod}^G_{\mathcal{M}}(\phi)$ signifies the sub-class of models in the class $\mathcal{M}$ that satisfy ϕ on all of their states.

Therefore, for some class of models $\mathcal{M}$ we have:

$$M *_{\mathcal{M}} T = \min_{\leq_M}(\mathrm{mod}_{\mathcal{M}}(T))$$

and its global version

$$M *^G_{\mathcal{M}} T = \min_{\leq_M}\left(\mathrm{mod}^G_{\mathcal{M}}(T)\right).$$

Accordingly, stopperedness has to be parameterised over local/global satisfaction. Definition 11 says that an ordering $\leq$ over $\mathcal{M}$ is stoppered over a collection of sets of models $\mathcal{C} \subseteq 2^{\mathcal{M}}$ iff, for each $X \in \mathcal{C}$ and any model $A \in X$, there exists a model $B \in X$ such that $B \leq A$ and B is $\leq$-minimal in X. The collection of sets $\mathcal{C}$ will eventually be defined as the collection of sets of models of some sentence or some set of sentences. As such, these collections will again be ultimately parameterised over global/local satisfaction. Further details will follow in the specific cases.

The questions described in Section 3 will now be reviewed in the modal case. Firstly, observe the model E in Figure 1. It essentially represents a system that starts from a p state (the diamond shape indicates

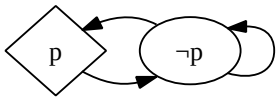

Figure 1. E, an example Kripke model.

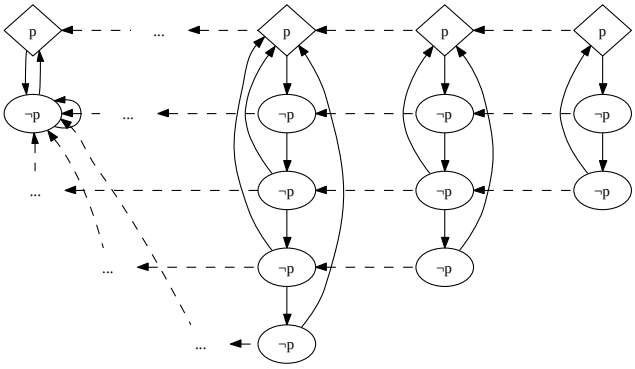

Figure 2. The model E (leftmost), and a chain of models that refine E and satisfy $\Box\Box\neg p$.

the starting state) and which will execute one or more transitions, the result of which is a $\neg p$ state, before returning to the p state.

The importance of the question of non-triviality is easily demonstrated: suppose we wanted to compute $E *_{\mathcal{K}} \Diamond p$ (note that this is the local version referring to the satisfaction of formulae at the initial state, over all Kripke models). It should be clear that a model that refines E and satisfies $\Diamond p$ does not exist, and as such, $E *_{\mathcal{K}} \Diamond p = \mathrm{mod}_{\mathcal{K}}(\Diamond p)$. The conditions under which this does not happen are thus of interest to us.

Suppose, instead, that we would like to compute $E *_{\mathrm{FIN}} \Box\Box\neg p$, where FIN denotes the class of finite Kripke models. In this case, it is easy to show that finite models that refine E and satisfy $\Box\Box\neg p$ indeed exist, as it can be seen in Figure 2. All the models shown in Figure 2 except for E, the leftmost, satisfy $\Box\Box\neg p$ on their initial states. Moreover, they form a chain, i.e. a set of models ordered totally by simulation (the dashed lines indicate the simulation relations). It is obvious that there exists an infinite number of finite models that are in between the leftmost model and the second one in terms of simulations, that satisfy $\Box\Box\neg p$ and are progressively smaller with respect to the ordering $\leq_E$.

Therefore, the question of stopperedness is important as a guarantee for the existence of minimal models. In this case it would guarantee that there exists a finite model M such that $E \leftarrow M$ (M refines E), $M \models \Box\Box\neg p$ (M satisfies the given property), and for any model N such that $E \leftarrow N \leftarrow M$ and $N \models \Box\Box\neg p$, then $M \leftarrow N$.

In fact, in this example, it is easy to find such a model and, moreover, one that is indeed the *minimum* of $\mathrm{mod}_{\mathrm{FIN}}(\Box\Box\neg p)$ with respect to $\leq_E$. This minimum model can be seen in Figure 3.

We will give an informal proof of the minimality of M. Observe Figure 4. The fact we are trying to prove is, essentially, that if there exists a model N with all the requirements, and is lower or equal to M then it is equal. More formally, if $N \models \Box\Box\neg p$ and $E \leftarrow N \leftarrow M$ then $N \rightarrow M$. A candidate model for N is shown in the middle of Figure 4 (for the purposes of readability some of the arrows of the simulation relations have been omitted).

First, observe that N must have a p-starting state, due to the fact that $N \leftarrow M$. From the same fact it follows that the starting state of N must have at least one $\neg p$-successor. Similarly, from $E \leftarrow N$ it follows that r_N cannot have a p-successor. Also, because of the requirement $N \models \Box\Box\neg p$, there cannot be any states

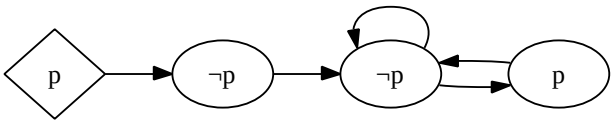

Figure 3. M, the minimum of $\mathrm{mod}_{\mathrm{FIN}}(\Box\Box\neg p)$ with respect to $\leq_E$.

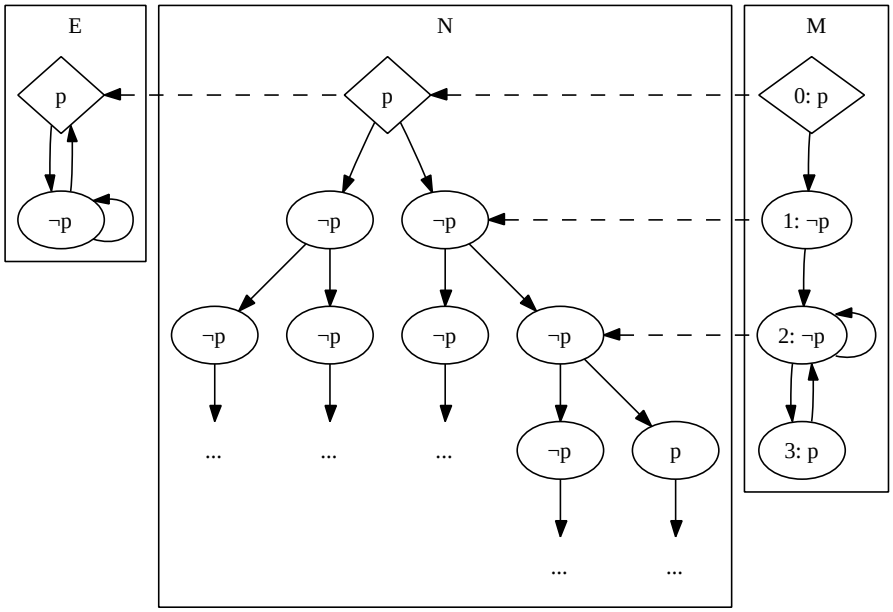

Figure 4. E, M and a candidate model N for $\mathrm{mod}_{\mathrm{FIN}}(\Box\Box\neg p)$, $E \leftarrow N \leftarrow M$ and $N \nrightarrow M$.

in N with distance 2 from r_N that have p. Therefore, N must look like what is shown in Figure 4 (modulo simulation equivalence).

Proving that M is the minimum model requires, essentially, the construction of a simulation from N to M. We will briefly define one such relation and argue that it is a simulation. The state r_N will of course be mapped to r_M. The level-1 and level-2 successors of N will be mapped to states 1 and 2 of M, respectively. For the remaining states, the mapping is defined according to the valuation: a $\neg p$-state of N will be mapped to state 2 and a p-state of N to state 3. The mapping of the $\neg p$ states is trivially adequate, since state 2 has both $\neg p$- and p-successors. The mapping of the p-state is slightly trickier in that state 3 has only a $\neg p$-successor. However, from the requirement that $E \leftarrow N$ follows that a p-state can only transition to a $\neg p$-state. Therefore, the defined relation is a simulation and as such M is the minimum.

This example is obviously a very simple one. We would like to be able to guarantee the existence of minimal models and be able to *compute* them, without recourse to a proof for each individual case. Therefore, it would be interesting to investigate the decidability of the problem of finding minimal models, as well as of reasoning about them.

4.3. m-Saturated models

Minimal refinement from the perspective of m-saturated models, $*_{\mathrm{MSAT}}$, will be discussed in this section. We will start with the definition of a few auxiliary notions and then proceed to address the issues of non-triviality and stopperedness.

In the following we will use the *ultrafilter extension* of a model. The internals of the construction are not relevant here, apart from two of its properties:

- the ultrafilter extension of a model M is another model $\mathrm{ue}(M)$ that is logically equivalent to M and,
- $\mathrm{ue}(M)$ is m-saturated.

Accounts of the construction appear in many places, e.g. [BdRV01].

A set of sentences T is called *complete* if for every sentence $\phi \in \mathcal{L}_{\mathrm{K}}$, either $T \vdash \phi$ or $T \vdash \neg\phi$. It is easy to see that if T is complete and consistent then there exists a model M such that $\mathrm{th}(M) = T$. But what happens when we restrict to the language of positive universal formulae? The following definition and lemma establish a criterion.

Definition 13. Let T be a set of sentences.

- T is *closed under taking disjuncts* iff whenever $\phi \vee \psi \in T$ then $\phi \in T$ or $\psi \in T$.
- T is *closed under $\mathcal{L}_{\mathrm{PU}}$-consequence* iff whenever $T \vdash \phi$ and $\phi \in \mathcal{L}_{\mathrm{PU}}$ then $\phi \in T$.

Lemma 3. Let $P \subseteq \mathcal{L}_{\mathrm{PU}}$. There exists a model M such that $P = \mathrm{PU}(M)$ if and only if P is consistent, closed under $\mathcal{L}_{\mathrm{PU}}$-consequence and taking disjuncts.

Proof. The left-to-right direction is trivial. Right-to-left: for a model M to have exactly P as its set of PU formulae, it must satisfy P and falsify its complement with respect to $\mathcal{L}_{\mathrm{PU}}$. In other words, there exists such a model iff $P, \overline{P^c} \nvdash \bot$. Assume the contrary. Then there exist formulae $\phi, \psi_1, \ldots, \psi_n$ such that $\phi \in P$ (note that P is closed under conjunction), $\neg\psi_i \in \overline{P^c}$ and $\phi, \neg\psi_1, \ldots, \neg\psi_n \vdash \bot$. But then, $\phi \vdash \psi_1 \vee \ldots \vee \psi_n$ and since P is closed under $\mathcal{L}_{\mathrm{PU}}$-consequence, $\psi_1 \vee \ldots \vee \psi_n \in P$. The set P is also closed under taking disjuncts so there exists $1 \leq j \leq n$ such that $\psi_j \in P$ which is a contradiction because $\psi_j \in P^c$. □

We may now turn to the first question about minimal refinement in MSAT, non-triviality.

Lemma 4. Let M be an m-saturated model and T a set of sentences. Then,

- there exists an m-saturated model N such that $M \leftarrow N$ and $N \models T$ iff $\mathrm{PU}(M) \cup T \nvdash \bot$.
- there exists an m-saturated model N such that $M \leftarrow N$ and $N \models_G T$ iff $\mathrm{PU}(M) \cup \Box^* T \nvdash \bot$.

Proof. We consider the local case first. Left-to-right: Since $M \leftarrow N$ it follows from Lemma 1 that $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$. Thus N is a model of both T and $\mathrm{PU}(M)$.

Right-to-left: Let N be a model of $\mathrm{PU}(M) \cup T$. Then, $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$. Since N may not be m-saturated, we take the ultrafilter-extension of N, $\mathrm{ue}(N)$ which is logically equivalent to N (and as such a model of $\mathrm{PU}(M) \cup T$) and m-saturated. It follows that $T \subseteq \mathrm{th}(\mathrm{ue}(N))$ and that $\mathrm{PU}(M) \subseteq \mathrm{PU}(\mathrm{ue}(N))$. As M is m-saturated it follows from Lemma 2 that $M \leftarrow \mathrm{ue}(N)$.

It is easy to see that the condition of non-triviality for the global case follows easily from the local one in view of the observation that $\mathrm{mod}^G_{\mathrm{MSAT}}(T) = \mathrm{mod}_{\mathrm{MSAT}}(\Box^* T)$. □

The following lemma and theorem concern stopperedness of the ordering over the class of m-saturated models. Lemma 5 enables us to apply Zorn's lemma by proving that, for any chain (a totally-ordered set of models), a suitable lower bound can be found, and indeed, the infimum.

Lemma 5. Let M be an m-saturated model and T a consistent set of sentences of which M is not a model. Let $C \subseteq \mathrm{mod}_{\mathrm{MSAT}}(T)$ be a non-empty chain with respect to $\leq_M$ where all of its members are simulated by M. Then there exists an m-saturated model of T which is the infimum of C (modulo simulation-equivalence).

Proof. Define $P = \bigcap_{N \in C} \mathrm{PU}(N)$. We will prove that there exists a model L with $\mathrm{PU}(L) = P$ which satisfies T. Since any model N in the chain is simulated by M, $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$ and therefore $\mathrm{PU}(M) \subseteq P$. Also, for any two models $N, N' \in C$ it will be the case that $\mathrm{PU}(N) \subseteq \mathrm{PU}(N')$ or $\mathrm{PU}(N') \subseteq \mathrm{PU}(N)$. The set P is obviously consistent as a subset of consistent sets. Also, it is easy to check that P is closed under $\mathcal{L}_{\mathrm{PU}}$-consequence.

We now prove that P is closed under taking disjuncts. Assume $\phi \vee \psi \in P$. Then, for all $N \in C$, $N \models \phi \vee \psi$. If all the models in C satisfy ϕ (or, respectively, ψ) then we are done, so assume that there exists a pair of models $N, N' \in C$ such that $N \models \phi \wedge \neg\psi$ and $N' \models \neg\phi \wedge \psi$. But this contradicts the fact mentioned above, that $\mathrm{PU}(N) \subseteq \mathrm{PU}(N')$ or $\mathrm{PU}(N') \subseteq \mathrm{PU}(N)$. Hence P is closed under taking disjuncts.

From Lemma 3 it follows that $P \cup \overline{P^c}$ is consistent. Assume that $P, \overline{P^c}, T \vdash \bot$. Then there exist formulae $\neg\phi_1, \ldots, \neg\phi_n \in \overline{P^c}$ such that $P, T, \neg\phi_1, \ldots, \neg\phi_n \vdash \bot$ or equivalently $P, T \vdash \phi_1 \vee \ldots \vee \phi_n$. Thus, for all $N \in C$, $N \models \phi_1 \vee \ldots \vee \phi_n$, hence $\phi_1 \vee \ldots \vee \phi_n \in \mathrm{PU}(N)$ and therefore $\phi_1 \vee \ldots \vee \phi_n \in P$. As P is closed under taking disjuncts there is one disjunct ϕ_j such that $\phi_j \in P$, which is a contradiction. Thus there is a model L of $P \cup \overline{P^c} \cup T$. The model L need not be m-saturated, but its ultrafilter extension $\mathrm{ue}(L)$ is, and as it is logically equivalent to L it will satisfy $P \cup \overline{P^c} \cup T$ too.

By the definition of P we have that for all $N \in C$, $\mathrm{PU}(\mathrm{ue}(L)) \subseteq \mathrm{PU}(N)$. Thus, by Lemma 2 we obtain $\mathrm{ue}(L) \leftarrow N$. Also, $\mathrm{PU}(M) \subseteq \mathrm{PU}(\mathrm{ue}(L))$ which implies that $M \leftarrow \mathrm{ue}(L)$. So, $\mathrm{ue}(L)$ is a lower bound of C with respect to $\leq_M$. In addition, for any other lower bound L' of C, it follows that $\mathrm{PU}(L') \subseteq \bigcap_{N \in C} \mathrm{PU}(N)$, or in other words, $\mathrm{PU}(L') \subseteq \mathrm{PU}(\mathrm{ue}(L))$, or, through Lemma 2, $L' \leftarrow \mathrm{ue}(L)$. Consequently, $L' \leq_M \mathrm{ue}(L)$ and thus that $\mathrm{ue}(L)$ is the infimum of C (modulo similarity). □

In Theorems 6 and 10 we prove stopperedness for the classes of m-saturated and finite models, respectively. The application of Zorn's lemma is a crucial part of such proofs. Since stopperedness concerns a slightly different set of conditions than the conclusion of Zorn's lemma, we provide and use an equivalent form of Zorn's lemma (Lemma 21), the proof of which can be found in the appendix.

In relation to stopperedness, we will consider two collections of sets of m-saturated models,

- $\mathcal{C}^{T}_{\mathrm{MSAT}} = \{\mathrm{mod}_{\mathrm{MSAT}}(S) \mid S \subseteq \mathcal{L}_{\mathrm{K}}\}$
- $\mathcal{C}^{GT}_{\mathrm{MSAT}} = \left\{\mathrm{mod}^{G}_{\mathrm{MSAT}}(S) \;\middle|\; S \subseteq \mathcal{L}_{\mathrm{K}}\right\}$

corresponding to local and global satisfaction respectively (T indicates that we consider sets of sentences instead of formulae). We can now turn to the question of stopperedness for MSAT.

Theorem 6. Let M be an m-saturated model and T a set of sentences. The ordering $\leq_M$ is stoppered over both $\mathcal{C}^{T}_{\mathrm{MSAT}}$ and $\mathcal{C}^{GT}_{\mathrm{MSAT}}$.

Proof. Firstly, we consider $\mathcal{C}^{T}_{\mathrm{MSAT}}$. Let T be a set of sentences. If $T \subseteq \mathrm{th}(M)$ then M is a *minimum* with respect to $\leq_M$ in $\mathrm{mod}_{\mathrm{MSAT}}(T)$, as well as any other m-saturated model N of T such that $M \leftrightarrows N$. It follows that for any m-saturated model N of T there is an m-saturated model of T, i.e. M, which is minimal and $M \leq_M N$.

In the case where $M \notin \mathrm{mod}_{\mathrm{MSAT}}(T)$, it may or may not be that $\mathrm{PU}(M) \cup T$ is consistent. If not, then by applying Lemma 4 it follows that there are no models in $\mathrm{mod}_{\mathrm{MSAT}}(T)$ that are simulated by M. Hence, only the third clause of the definition of $\leq_M$ can ever apply, rendering all (simulation-distinct) models in $\mathrm{mod}_{\mathrm{MSAT}}(T)$ incomparable. In this case, for any model $N \in \mathrm{mod}_{\mathrm{MSAT}}(T)$ there is a model N' (namely N itself) such that $N' \leq_M N$, where N' is minimal.

Thus, we assume that $\mathrm{PU}(M) \cup T$ is consistent. Because of the second clause of the definition of the ordering, it is easy to see that in this case the set of minimal elements will be a subset of $\mathrm{mod}_{\mathrm{MSAT}}(\mathrm{PU}(M) \cup T)$. Therefore we restrict our attention to the models in $\mathrm{mod}_{\mathrm{MSAT}}(\mathrm{PU}(M) \cup T)$ which, by virtue of Lemma 4, are all simulated by M.

Then, for any non-empty chain C in $\mathrm{mod}_{\mathrm{MSAT}}(\mathrm{PU}(M) \cup T)$ Lemma 5 applies, yielding a lower bound of C within $\mathrm{mod}_{\mathrm{MSAT}}(\mathrm{PU}(M) \cup T)$. Therefore by Zorn's lemma, for any model $N \in \mathrm{mod}_{\mathrm{MSAT}}(T)$ there exists another model $N' \in \mathrm{mod}_{\mathrm{MSAT}}(T)$ such that N' is minimal and $N' \leq_M N$. Thus $\leq_M$ is stoppered over $\mathcal{C}^{T}_{\mathrm{MSAT}}$.

For the case of $\mathcal{C}^{GT}_{\mathrm{MSAT}}$ we need only observe that since $\mathrm{mod}^{G}_{\mathrm{MSAT}}(T) = \mathrm{mod}_{\mathrm{MSAT}}(\Box^{*}T)$, stopperedness over $\mathcal{C}^{GT}_{\mathrm{MSAT}}$ is reduced to stopperedness over $\mathcal{C}^{T}_{\mathrm{MSAT}}$. □

To summarise the set of results on m-saturated models,

- For any set of modal sentences T and an m-saturated model M, the minimal refinement $M *^{[G]}_{\mathrm{MSAT}} T$ is non-trivial (in the sense that the third clause of Definition 9 for $\leq_M$ is *not* the only one that applies within $\mathrm{mod}^{[G]}_{\mathrm{MSAT}}(T)$), if and only if the set $\mathrm{PU}(M) \cup T$ is consistent, in the local case, or $\mathrm{PU}(M) \cup \Box^{*}T$ in the global case.
- For any m-saturated model M and any set of sentences T,

$$M *^{[G]}_{\mathrm{MSAT}} T = \min_{\leq_M} \left(\mathrm{mod}^{[G]}_{\mathrm{MSAT}}(T)\right) \neq \emptyset.$$

This set of results will form the basis of the corresponding ones in the finite case. This will, in general, be achieved by exploiting the fact that finite models are m-saturated, and then by constructing finite models that lie 'below' (in terms of $\leq_M$) the m-saturated ones provided by Lemma 4 and Theorem 6.

4.4. Finite models

We will now turn to the treatment of minimal refinement in the finite case. As already mentioned, the class of all finite Kripke models will be denoted by FIN.

Let M be a model and s, t states in M. A *path* from s to t is a finite sequence of states of M such that the first state is s, the last is t and for any pair of states s_i, s_{i+1} in the sequence, there exists a j such that $(s_i, s_{i+1}) \in R^{j}_{M}$. The *depth* of a state s is defined as the minimum length of a path from the root to s if such

a path exists, otherwise as ω. A state s is said to have *in-degree one* whenever it has a unique ancestor with respect to the union of all accessibility relations in M. A model M will be called *smooth* iff every state in W_M apart from the root has in-degree one and finite depth. For every model M there is a smooth one M^s, known also as the *unfolding* of M, such that $M \sim M^s$. The proof of this result as well as of a general version of the following lemma can be found in [dR95]. This lemma will allow us, in what follows, to concentrate on functional simulations instead of arbitrary ones.

Lemma 7. Let N and M be models such that N is smooth, M is m-saturated and $M \leftarrow N$. Then there exists a functional simulation from N to M.

Proof. We will define a function $S : W_N \to W_M$ and prove by induction that for any $t \in W_N$, $\mathrm{PE}(t) \subseteq \mathrm{PE}(S(t))$ (S can then be proved to be a simulation in a manner identical to the one presented in Lemma 2.) We set $S(r_N) = r_M$. Since $M \leftarrow N$ it follows from Lemma 1 that $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$ and thus $\mathrm{PU}(S(r_N)) \subseteq \mathrm{PU}(r_N)$, or $\mathrm{PE}(r_N) \subseteq \mathrm{PE}(S(r_N))$.

Assume that S has been defined for all states in N of depth up to $n-1$ and let $t \in W_N$ be a state of depth n. Since N is smooth, t has a uniquely defined ancestor t' with respect to some relation R^i_N. By the inductive hypothesis, $\mathrm{PE}(t') \subseteq \mathrm{PE}(S(t'))$. So, for any finite set of PE sentences $F \subseteq \mathrm{PE}(t)$, it follows that $t' \models \Diamond_i \bigwedge F$, hence $S(t') \models \Diamond_i \bigwedge F$, and as such, there exists a $u \in W_M$ such that $u \models \bigwedge F$ and $(S(t'), u) \in R^i_M$. In other words, $\mathrm{PE}(t)$ is finitely satisfiable on the R^i_M-successors of $S(t')$ which through the m-saturation of M gives us that $\mathrm{PE}(t)$ is satisfiable at a R^i_M-successor u'. We set $S(t) = u'$ and this completes the proof. □

As explained in the end of the previous section, we plan to use the results on m-saturated models as a basis for the corresponding ones in the finite case. This, in general, boils down to the following scenario: given a finite model M and a sentence $\phi \in \mathcal{L}_{\mathrm{K}}$, we are interested in some property of $M *^{[G]}_{\mathrm{FIN}} \phi$. Since M is a finite model, it is also an m-saturated one, allowing us to apply one of the results of Section 4.3, yielding an m-saturated model N such that $M \leftarrow N$ and $N \models_{[G]} \phi$. All that is needed in order to obtain the corresponding result in the finite case is to construct a finite model L such that $M \leftarrow L \leftarrow N$ and $L \models_{[G]} \phi$. The following definition and lemma will give us exactly such a tool.

Definition 14. Let Σ be a set of sentences such that $\mathcal{A} \subseteq \Sigma$, M a model and E an equivalence relation on W_M.

1. Σ is *subformula-closed* iff for every ϕ, ψ,
   - $\neg\phi \in \Sigma$ implies $\phi \in \Sigma$,
   - $\phi \wedge \psi \in \Sigma$ implies $\phi \in \Sigma$ and $\psi \in \Sigma$ and
   - $\Diamond_i\phi \in \Sigma$ implies $\phi \in \Sigma$, for all $1 \leq i \leq m$.

   (in the unabbreviated language). The set of subformulae of a formula ϕ will be the smallest set of formulae that contains ϕ and is closed under subformulae.
2. We define an equivalence relation $\equiv_{\Sigma,E}$ on W_M as follows.

   $$s \equiv_{\Sigma,E} t \quad \text{iff} \quad (\forall \phi \in \Sigma, s \models \phi \Leftrightarrow t \models \phi) \wedge (s,t) \in E.$$

   We denote the $\equiv_{\Sigma,E}$-equivalence class of s by $[s]$.
3. We define a model M_f as follows:
   - $W_{M_f} = \{[s] \mid s \in W_M\}$.
   - $r_{M_f} = [r_M]$.
   - For all $s, t \in W_M$, $([s],[t]) \in R^i_{M_f}$ if there exist states $s', t' \in W_M$ such that $s' \in [s], t' \in [t]$ and $(s', t') \in R^i_M$.
   - $p \in v_{M_f}([s])$ iff $p \in v_M(s)$ for all $p \in \mathcal{A}$.

It is not hard to see that if Σ is subformula-closed then the model M_f defined above is a slightly modified *filtration* of M (see, e.g. [Che80, BdRV01]). As such, it has similar properties to a filtration:

- Let n be the number of E-equivalence classes in M. If n and $|\Sigma|$ are finite, then M_f is finite and $|M_f| \leq 2^{|\Sigma|} \cdot n$.
- For all formulae $\psi \in \Sigma$ and states $s \in W_M$, $M, s \models \psi$ iff $M_f, [s] \models \psi$.

Lemma 8. Let M be a finite model, L a possibly infinite model such that $M \leftarrow L$ and ϕ a formula. Then there exists a finite model N such that $M \leftarrow N \leftarrow L$ and that $N \models_{[G]} \phi$ iff $L \models_{[G]} \phi$. Moreover, $|N| \leq 2^{|\phi|+2\cdot|\mathcal{A}|} \cdot |M|$.

Proof. Define Σ as the set of subformulae of ϕ unioned with the set of atomic propositions $\mathcal{A}$, along with its atoms negated $\overline{\mathcal{A}}$. Let $K = L^s$ be the smooth counterpart of L. Since $L \sim K$ and $M \leftarrow L$ it follows that $M \leftarrow K$ (the composition of a bisimulation and a simulation yields a simulation). Moreover, since M is finite it is also m-saturated, thus Lemma 7 applies, giving us a functional simulation S between K and M. By $S(t)$ we denote the (unique) state $t' \in W_M$ such that $(t, t') \in S$ for some state $t \in W_K$. Using S we define an equivalence relation E on W_K as

$$(s, t) \in E \quad \text{iff} \quad s, t \in W_K \quad \text{and} \quad S(s) = S(t).$$

Using K, Σ and E we define $N = K_f$ and the associated $[\cdot]$ mapping. Thus, it follows that

- Σ is finite by definition and has a size of at most $|\phi| + 2 \cdot |\mathcal{A}|$. Since S maps K into M, a finite model, the number of E-equivalence classes must be finite and at most $|M|$. Therefore, N is finite and $|N| \leq 2^{|\phi|+2\cdot|\mathcal{A}|} \cdot |M|$.
- N is a filtration of K and as such the usual properties hold, i.e. $N \models_{[G]} \phi$ iff $K \models_{[G]} \phi$ iff $L \models_{[G]} \phi$ (for the global case, note that $[\cdot]$ is surjective on W_N).

We now prove that the necessary simulations exist between M, N, L.

$N \leftarrow L$: We first prove that $N \leftarrow K$. Since $L \sim K$ it then follows that $N \leftarrow L$ as well. Define a relation $Q \subseteq W_K \times W_N$ such that

$$(s, t) \in Q \quad \text{iff} \quad t = [s].$$

Essentially, Q is an embedding of K in N. We prove that Q is a simulation:

- Obviously, $(r_K, r_N) \in Q$ since $r_N = [r_K]$.
- Also, $(s, [s]) \in Q$ implies $v_K(s) = v_N([s])$ by the definition of v_K and from the fact that Σ contains all the propositional letters.
- If $(s, t) \in R^i_K$ then by definition there is an R^i_N-successor of $[s]$ that is the image of t, namely $[t]$.

$M \leftarrow N$: Define a relation $U \subseteq W_N \times W_M$ as

$$(t, u) \in U \quad \text{iff} \quad \exists s \in W_K \quad \text{such that} \quad [s] = t \wedge S(s) = u.$$

We prove that U is a simulation.

- By definition, $[r_K] = r_N$. Since S is a simulation, $S(r_K) = r_M$. Thus, $(r_N, r_M) \in U$.
- Suppose that $(t, u) \in S$. Thus, there exists a state s in K such that $[s] = t$, therefore $v_K(s) = v_N(t)$ given that $\mathcal{A} \cup \overline{\mathcal{A}} \subseteq \Sigma$. Moreover, $S(s) = u$, thus $v_K(s) = v_M(u)$. Therefore, $v_N(t) = v_M(u)$.
- Assume that $(t_1, t_2) \in R^i_N$. That means that there exist two states s_1, s_2 in K such that $[s_1] = t_1, [s_2] = t_2$ and $(s_1, s_2) \in R^i_K$. As such, since S is a simulation, $(S(s_1), S(s_2)) \in R^i_M$. Thus, $(t_1, S(s_1)) \in U$ and $(t_2, S(s_2)) \in U$.

□

With the help of this result we may approach the questions on minimal refinement in the finite case. We start from non-triviality.

Lemma 9. Let ϕ be a formula and M a finite model. Then,

- there exists a finite model N such that $N \models \phi$ and $M \leftarrow N$ iff $\mathrm{PU}(M), \phi \not\vdash \bot$.
- there exists a finite model N such that $N \models_G \phi$ and $M \leftarrow N$ iff $\mathrm{PU}(M) \cup \Box^*\phi \not\vdash \bot$.

Moreover, it is decidable whether such a model exists.

Proof. Left-to-right: In the local case, $N \models \phi$. Since $M \leftarrow N$, it follows that $\mathrm{PU}(M) \subseteq \mathrm{PU}(N)$ so it cannot possibly be the case that $\mathrm{PU}(M)$ is inconsistent with ϕ. The global case follows similarly by observing that $N \models_G \phi$ iff $N \models \Box^*\phi$.

Right-to-left: We address the local and global case simultaneously. Since $\mathrm{PU}(M) \cup \phi$ is consistent ($\mathrm{PU}(M) \cup \Box^* \phi$), from Lemma 4 it follows that there exists an m-saturated model N' such that $N' \models \phi$ ($N' \models \Box^* \phi$ or equivalently $N' \models_G \phi$), and $M \leftarrow N'$. But then, by Lemma 8 there exists a finite model $N \models \phi$ ($N \models_G \phi$) such that $M \leftarrow N \leftarrow N'$ and $|N| \leq |M| \cdot 2^{|\phi|+|\mathcal{A}|}$.

Since the size of N is bounded we can enumerate all the models that satisfy (globally satisfy) ϕ with up to $|M| \cdot 2^{|\phi|+|\mathcal{A}|}$ states and check whether any of these is simulated by M. Thus the problem is decidable. □

As with m-saturated models, we will again consider two collections of sets of finite models for stopperedness:

- $\mathcal{C}^{\phi}_{\mathrm{FIN}} = \{\mathrm{mod}_{\mathrm{FIN}}(\phi) \mid \phi \in \mathcal{L}_{\mathrm{K}}\}$
- $\mathcal{C}^{G\phi}_{\mathrm{FIN}} = \left\{\mathrm{mod}^{G}_{\mathrm{FIN}}(\phi) \;\middle|\; \phi \in \mathcal{L}_{\mathrm{K}}\right\}$

We will address the issue of stopperedness for FIN for these two classes.

Theorem 10. Let M be a finite model. The ordering $\leq_M$ is stoppered over $\mathcal{C}^{\phi}_{\mathrm{FIN}}$ and $\mathcal{C}^{G\phi}_{\mathrm{FIN}}$.

Proof. As in the proof of Theorem 6, it is easy to check that when $M \models_{[G]} \phi$ or $\mathrm{PU}(M), \phi \vdash \bot$ ($\mathrm{PU}(M) \cup \Box^* \phi \vdash \bot$) then for any model $N \in \mathrm{mod}^{[G]}_{\mathrm{FIN}}(\phi)$ there exists a model $N' \in \mathrm{mod}^{[G]}_{\mathrm{FIN}}(\phi)$ such that $N' \leq_M N$ and N' is minimal. So we assume that $M \not\models_{[G]} \phi$, that $\mathrm{PU}(M), \phi \nvdash \bot$ ($\mathrm{PU}(M) \cup \Box^* \phi \nvdash \bot$) and restrict our attention to the models in $\mathrm{mod}_{\mathrm{FIN}}(\mathrm{PU}(M) \cup \{\phi\})$ ($\mathrm{mod}_{\mathrm{FIN}}(\mathrm{PU}(M) \cup \Box^* \phi)$).

Let $C \subseteq \mathrm{mod}_{\mathrm{FIN}}(\mathrm{PU}(M) \cup \{\phi\})$ ($C \subseteq \mathrm{mod}_{\mathrm{FIN}}(\mathrm{PU}(M) \cup \Box^* \phi)$) be a chain with respect to $\leq_M$. Since finite models are m-saturated, from Theorem 5 we obtain that there is an m-saturated model L which satisfies $\mathrm{PU}(M) \cup \{\phi\}$ ($\mathrm{PU}(M) \cup \Box^* \phi$) and is a lower bound of C with respect to $\leq_M$. But then, by Lemma 8, there is a finite model N such that $N \models_{[G]} \phi$ and $M \leftarrow N \leftarrow L$. Therefore, N is a lower bound of C and by applying Zorn's lemma we obtain stopperedness for the class of finite models. □

We continue with a set of decidability results concerning the finite case. Firstly we prove that checking whether a specific model N is minimal with respect to a model M and ϕ, i.e. whether $N \in M *^{[G]}_{\mathrm{FIN}} \phi$, is decidable.

Lemma 11. Let M, N be finite models and ϕ a sentence. There is an effective procedure for deciding $N \in M *^{[G]}_{\mathrm{FIN}} \phi$.

Proof. We first test for non-triviality via Lemma 9. If $M *^{[G]}_{\mathrm{FIN}} \phi$ is trivial then we check whether $N \models_{[G]} \phi$. In this case, N is minimal iff $N \models_{[G]} \phi$.

Thus, we assume that $M *^{[G]}_{\mathrm{FIN}} \phi$ is non-trivial. Then, we check whether $M \leftarrow N$. If not, then surely N is not minimal. So, we assume that $M \leftarrow N$. Now, N is minimal iff there is no other model $N' \in \mathrm{mod}^{[G]}_{\mathrm{FIN}}(\phi)$ such that $N' <_M N$. Assume N is not minimal. Then there exists $N' \in \mathrm{mod}^{[G]}_{\mathrm{FIN}}(\phi)$ such that $M \leftarrow N' \leftarrow N$ but $N' \nrightarrow N$ (by the definition of the ordering and the assumption that $M \leftarrow N$ it follows that if there exists $N' <_M N$, it will have to be due to the first clause of the definition). By applying Lemma 8 to the pair of models M, N' it follows that there exists a model $N'' \in \mathrm{mod}^{[G]}_{\mathrm{FIN}}(\phi)$ such that $M \leftarrow N'' \leftarrow N'$ and thus, $N'' \nrightarrow N$. Therefore, N is not minimal iff there exists a model N'' of ϕ which is strictly smaller than N with respect to $\leq_M$ and it has at most $|M| \cdot 2^{|\phi|+|\mathcal{A}|}$ states.

Consequently, given N, M and ϕ, we can enumerate all the models that satisfy ϕ (globally satisfy) and have up to $|M| \cdot 2^{|\phi|+|\mathcal{A}|}$ states, of which there is a finite number. For each model L we check the simulations $M \leftarrow L$, $L \leftarrow N$, $L \nrightarrow N$. There exists such a model iff $N \notin M *^{[G]}_{\mathrm{FIN}} \phi$. □

The next theorem characterises the structure of $M *^{[G]}_{\mathrm{FIN}} \phi$ with respect to the ordering. It asserts that each $\leq_M$-equivalence class of models in $M *^{[G]}_{\mathrm{FIN}} \phi$ contains a representative model of bounded size.

Theorem 12. Let M be a finite model and ϕ a formula such that $M *^{[G]}_{\mathrm{FIN}} \phi$ is non-trivial. Then, there is a finite, computable set of finite models $\Delta^{[G]}_{M,\phi} \subseteq M *^{[G]}_{\mathrm{FIN}} \phi$ with the property that for any model $N \in M *^{[G]}_{\mathrm{FIN}} \phi$ there is a model N' such that

- $N' \in \Delta^{[G]}_{M,\phi}$,
- $N \leftrightarrows N'$ and
- $|N'| \leq |M| \cdot 2^{|\phi|+|\mathcal{A}|}$.

Proof. Let $N \in M *^{[G]}_{\text{FIN}} \phi$. The application of Lemma 8 gives us a model N' of ϕ such that $M \leftarrow N' \leftarrow N$ and $|N'| \leq |M| \cdot 2^{|\phi|+|\mathcal{A}|}$. But since N is minimal, it follows that $N \leftrightarrows N'$. Thus $\Delta^{[G]}_{M,\phi}$ can be computed by enumerating the finite models of ϕ that have at most $|M| \cdot 2^{|\phi|+|\mathcal{A}|}$ states and checking them for minimality via Lemma 11. □

A corollary of the above is that if for some finite model M and a formula ϕ, $M *^{[G]}_{\text{FIN}} \phi$ is non-trivial, then the number of simulation-equivalence classes of finite models that constitute $M *^{[G]}_{\text{FIN}} \phi$ is finite. We will now examine the decidability of reasoning about the results of the operation.

Theorem 13. Assume that $M *^{[G]}_{\text{FIN}} \phi$ is non-trivial. Let ψ be a formula in $\mathcal{L}_{\text{PU}} \cup \mathcal{L}_{\text{PE}}$. Then, there is an effective procedure for deciding $M *^{[G]}_{\text{FIN}} \phi \models_{[G]} \psi$ (the global and local cases can be combined, i.e. $M *_{\text{FIN}} \phi \models_G \psi$).

Proof. Consider the partitioning of $M *^{[G]}_{\text{FIN}} \phi$ by simulation-equivalence: in view of the previous result, there is a (finite) number of simulation-equivalence classes of finite models in $M *^{[G]}_{\text{FIN}} \phi$. Let $E \subseteq M *^{[G]}_{\text{FIN}} \phi$ be such an equivalence class. For any two models $N_1, N_2 \in E$ it holds that $\text{PU}(N_1) = \text{PU}(N_2)$ and equivalently $\text{PE}(N_1) = \text{PE}(N_2)$. In other words, the problem of checking whether all models in $M *^{[G]}_{\text{FIN}} \phi$ satisfy ψ (locally/globally), where $\psi \in \mathcal{L}_{\text{PU}} \cup \mathcal{L}_{\text{PE}}$, reduces to checking whether for every simulation-equivalence class E in $M *^{[G]}_{\text{FIN}} \phi$, there is a model $N \in E$ such that $N \models_{[G]} \psi$. But $\Delta^{[G]}_{M,\phi}$ contains at least one model from each such equivalence class, so the problem is further reduced to whether $\Delta^{[G]}_{M,\phi} \models_{[G]} \psi$ or not. Since $\Delta^{[G]}_{M,\phi}$ is finite and computable, and the problem of checking (global/local) satisfaction of a formula on a finite model is decidable, the overall problem is also decidable. □

Indeed, it is easy to see that the query language can be any language that is preserved under simulation-equivalence, e.g. ACTL.

4.5. Related work

Van der Meyden, in [Mey91], describes a framework designed to support artificial intelligence models of legal reasoning. Specifically, a logic for deontic action specification is developed, and proved to be complete with respect to the semantics presented therein. This logic is related to the framework presented here in that a process of minimisation is central to the model theory of the language, and that part of the ordering relevant to this minimisation is simulation. However, the approaches still differ significantly. Firstly, the work presented here is geared towards reasoning about specifications and processes rather than a framework designed for AI reasoning. Secondly, van der Meyden's work is closer to logic programming than this research, in that the minimisation is employed to provide unique models of theories (minima), rather than to perform changes to a theory or a model. Indeed, van der Meyden's effort is concentrated in ensuring the completeness of the proof theory with which derivation can be performed, rather than using the models as possible representations of systems. Lastly, the language considered is a clausal one, i.e. one that only allows reasoning about rules that contain negation only in their heads, and not in their bodies.

5. Minimal refinement in temporal logic

5.1. Introduction

Modal logic is a basic language for expressing specifications and describing processes. However, it is limited in what kinds of specifications it can express. For example, consider the case whereby a designer wants to minimally refine a system by a property of the form 'in all possible states of the system, if the atomic

variable Request is true then eventually RequestGranted becomes true'. This property dictates that for every state s that satisfies Request, there is a finite path i starting at s and ending at a state t, such that $t \models$ RequestGranted. It is easy to see that a modal formula ϕ cannot express such a property as it can only impose restrictions on states whose distance from s is at most as great as the nesting depth of the modalities in ϕ; since that depth is finite, restrictions on states whose distance from s is greater than that depth cannot be enforced and therefore an artificial bound is put on the time the variable RequestGranted is satisfied.

To address this lack of expressiveness we will instantiate the framework of minimal refinement for a temporal logic. We chose the logic ACTL [GL94], a fragment of CTL [CES86], a branching-time temporal logic well-known for its use in model checking. ACTL is a definite improvement on modal logic, since it provides temporal operators such as *until* and *eventually*. It is also incomparable with modal logic, since it is only universally quantified; for example, the modal property 'there is a next state that makes the atomic variable p true' cannot be expressed in ACTL. Finally, ACTL is an expressive language that is preserved by (fair) simulation, a fact that leads to a set of results on minimal refinement obtained in a straightforward way.

In the following we will also do away with the distinction between local and global minimal refinement; a formula of the form *globally-*ϕ (AGϕ) immediately implies the satisfaction of ϕ on all of the (reachable) states of a model, rendering unnecessary the existence of two versions of the operation.

The outline of this section is as follows. Minimal refinement with ACTL as the object language and transition systems with fairness constraints is examined in Section 5.2. The issues of non-triviality, stopperedness and decidability are discussed in the same section. Finally, an implementation is developed and a case study examined in Section 5.3.

5.2. Minimal refinement in the temporal case

In this section, *fair simulation* is used to instantiate the notion of refinement. Thus, the ordering $\leq_M$ will now be understood to involve fair simulations where it makes use of refinements ($\leftarrow$). The operation of minimal refinement in this context is

$$M *_{\mathrm{FTSF}} \phi = \min_{\leq_M}(\mathrm{mod}_{\mathrm{FTSF}}(\phi))$$

where $M \in \mathcal{M}_{\mathrm{FTSF}}$ and $\phi \in \mathcal{L}_{\mathrm{ACTL}}$.

Of course, the problem of triviality applies in this case too. It is easy to see that if a ACTL formula does not force an inconsistent initial state (e.g. a formula like $p \wedge \neg p$), then it has a very simple model: one that only contains a few initial states and no transitions at all. It is also easy to verify that such a model refines any other model that is compatible with it, in terms of its valuation of its initial states (note that such a model satisfies almost all ACTL formulae: the ones it does not are of the form $\phi_p \wedge \psi$ where ϕ_p is a propositional formula which is false on its initial states). The next lemma formalises this intuition, which incidentally demonstrates that in the context of ACTL, non-triviality is a less interesting issue. For the purposes of the next lemma, a formula that is formed by propositional atoms in some set $\mathcal{A}$, and the connectives $\neg, \wedge, \vee$, will be called a *propositional formula on* $\mathcal{A}$.

Lemma 14. Let M be a model in FTSF and ϕ a satisfiable formula of ACTL. The minimal refinement $M *_{\mathrm{FTSF}} \phi$ is non-trivial iff there exists a state $s \in S_M$ such that for any propositional formula ψ on $\mathcal{A}_M$, $\phi \models \psi$ implies that $M, s \models \psi$.

Proof. Left-to-right: Assume that $M *_{\mathrm{FTSF}} \phi$ is non-trivial; then, there exists a model N such that $N \models \phi$ and $M \leftarrow N$. So, pick any state $t \in S_N$ and its image $s \in S_M$ under the witnessing fair simulation for $M \leftarrow N$. Also, pick any propositional formula ψ on $\mathcal{A}_M$, which by the fact that $\mathcal{A}_M \subseteq \mathcal{A}_N$ (by Definition 8 and $M \leftarrow N$), is a propositional formula on $\mathcal{A}_N$. If $\phi \models \psi$ then $N \models \psi$ (because $N \models \phi$), i.e., for all states $t' \in S_N$, $N, t' \models \psi$ and therefore $N, t \models \psi$. But ψ is also a ACTL formula, and as such, it is preserved by fair simulation. Therefore, $M, s \models \psi$.

Right-to-left: Given $s \in S_M$ with the required assumptions, define a model N such that $W_N = S_N = \{t\}$, $R_N = \mathcal{F}_N = \emptyset$, $\mathcal{A}_N = \mathcal{A}_M$ and $v_N(t) = v_M(s)$. It is easy to see that, trivially, $M \leftarrow N$. Now, assume that $N, t \not\models \phi$. Because of the absence of any fair path in N starting at T, it must be that the valuation of t is the one that falsifies ϕ, something contradictory with the assumption. □

We will proceed with a few lemmas that will form the basis for the rest of the results on $*_{\mathrm{FTSF}}$. The definition of the synchronous product of two models in $\mathcal{M}_{\mathrm{FTSF}}$ follows, and is similar to the natural one, plus the item that defines the fairness constraints.

Definition 15. Given two models A, B define the *synchronous product* $A \times B$ to be a model with the following parts.

- $W_{A\times B} = \{(a,b) \mid a \in W_A,\ b \in W_B,\ v_A(a) \cap \mathcal{A}_B = v_B(b) \cap \mathcal{A}_A\}$,
- $S_{A\times B} = (S_A \times S_B) \cap W_{A\times B}$,
- $\mathcal{A}_{A\times B} = \mathcal{A}_A \cup \mathcal{A}_B$,
- $v_{A\times B}((a,b)) = v_A(a) \cup v_B(b)$,
- $(a,b) \to_{A\times B} (a',b')$ iff $a \to_A a'$ and $b \to_B b'$,
- $\mathcal{F}_{A\times B} = \{(P \times W_B) \cap W_{A\times B} \mid P \in \mathcal{F}_A\} \cup \{(W_A \times Q) \cap W_{A\times B} \mid Q \in \mathcal{F}_B\}$.

As noted, the definition of the product structure is the natural one: by matching the common parts of the valuation, we construct composite states. Accordingly, the transition relation is the restriction of the cartesian product of the transition relations of the factors, on the resulting state space of the product. As expected, it is easy to prove that $A \leftarrow A \times B$ and $B \leftarrow A \times B$ for any A and B. The following result, proved in [GL94], asserts that a sequence of states is a fair path in the product if and only if its projections are fair paths in the product's factors, and the corresponding pairs of states are states in the product.

Lemma 15. Let A, B be models. The following conditions are equivalent.

1. $\pi_{A\times B} = (a_0,b_0)(a_1,b_1)\ldots$ is a fair path in $A \times B$.
2. $\pi_A = a_0a_1\ldots$ and $\pi_B = b_0b_1\ldots$ are fair paths in A, B respectively and for all i, $(a_i,b_i) \in W_{A\times B}$.

The next lemma asserts that the product defined above enjoys one of the universal properties of products, that is, if a model is simulated by both of the factors of a product, then it is simulated by the product as well.

Lemma 16. Let A, B, C be models such that $B \leftarrow A$ and $C \leftarrow A$. Then, $B \times C \leftarrow A$.

Proof. Let H_B, H_C be the witnessing fair simulations from A to B and C respectively. Define a relation $H_{B\times C} \subseteq W_A \times W_{B\times C}$,

$$(a,(b,c)) \in H_{B\times C} \quad \text{iff} \quad a \in W_A, (b,c) \in W_{B\times C}, (a,b) \in H_B, (a,c) \in H_C.$$

We will prove that $H_{B\times C}$ is a fair simulation from A to $B \times C$. First observe that by assumption $\mathcal{A}_A \supseteq \mathcal{A}_B$ and $\mathcal{A}_A \supseteq \mathcal{A}_C$ so $\mathcal{A}_A \supseteq \mathcal{A}_B \cup \mathcal{A}_C = \mathcal{A}_{A\times B}$.

1. For any state $\alpha \in S_A$ there is a state $(\beta,\gamma) \in S_{B\times C}$ such that $(\alpha,(\beta,\gamma)) \in H_{B\times C}$:
   From the fact that H_B, H_C are fair simulations it follows that there exist states $\beta \in S_B$ and $\gamma \in S_C$ such that $(\alpha,\beta) \in H_B$ and $(\alpha,\gamma) \in H_C$. From the same fact it follows that $v_A(\alpha) \cap \mathcal{A}_B = v_B(\beta)$ and $v_A(\alpha) \cap \mathcal{A}_C = v_C(\gamma)$ so $v_B(\beta) \cap \mathcal{A}_C = v_C(\gamma) \cap \mathcal{A}_B$ and as such $(\beta,\gamma) \in W_{B\times C}$. Moreover, $(\beta,\gamma) \in S_{B\times C}$. From the definition of $H_{B\times C}$ it follows that $(\alpha,(\beta,\gamma)) \in H_{B\times C}$.
2. For all $a \in W_A$ and $(b,c) \in W_{B\times C}$, $(a,(b,c)) \in H_{B\times C}$ implies that
   (a) $v_A(a) \cap \mathcal{A}_{B\times C} = v_{B\times C}(b,c)$:
   As in the previous item, from $(a,(b,c)) \in H_{B\times C}$ we get that $v_A(a) \cap \mathcal{A}_B = v_B(b)$ and $v_A(a) \cap \mathcal{A}_C = v_C(c)$. Thus $v_A(a) \cap \mathcal{A}_{B\times C} = v_A(a) \cap (\mathcal{A}_B \cup \mathcal{A}_C) = (v_A(a) \cap \mathcal{A}_B) \cup (v_A(a) \cap \mathcal{A}_C) = v_B(b) \cup v_C(c) = v_{B\times C}(b,c)$.
   (b) For every fair path $\pi_A = a_0a_1\ldots$ in A with $a_0 = a$ there exists a fair path $\pi_{B\times C} = (b_0,c_0)(b_1,c_1)\ldots$ in $B \times C$ with $(b_0,c_0) = (b,c)$ such that for every i, $(a_i,(b_i,c_i)) \in H_{B\times C}$:
   From the fact that H_B, H_C are fair simulations we obtain that there exist fair paths $\pi_B = b_0, b_1, \ldots$ and $\pi_C = c_0, c_1, \ldots$ in B, C respectively, such that $b_0 = b$, $c_0 = c$ and for all i, $(a_i,b_i) \in H_B$ and $(a_i,c_i) \in H_C$.
   Since $(a_i,b_i) \in H_B$ and $(a_i,c_i) \in H_C$, it follows that $v_A(a_i) \cap \mathcal{A}_B = v_B(b_i)$ and $v_A(a_i) \cap \mathcal{A}_C = v_C(c_i)$, thus $v_B(b_i) \cap \mathcal{A}_C = v_C(c_i) \cap \mathcal{A}_B$. As such, $(b_i,c_i) \in W_{B\times C}$. From Lemma 15 it follows that $\pi_{B\times C}$ is a fair path in $B \times C$. Moreover, from the definition of $H_{B\times C}$ it follows that $(a_i,(b_i,c_i)) \in H_{B\times C}$.

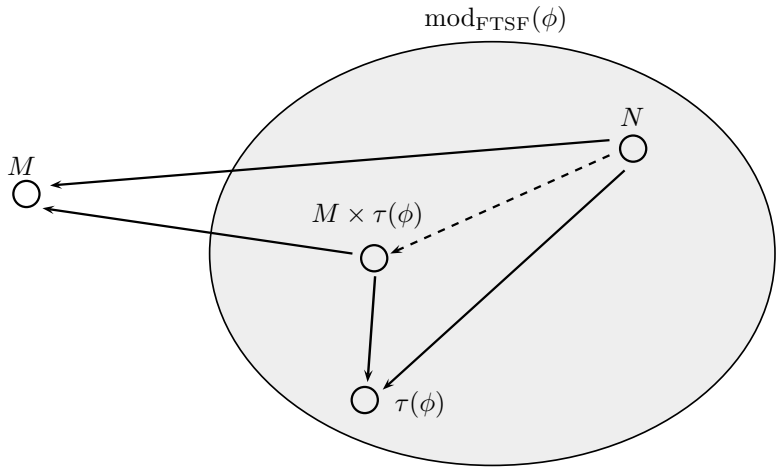

Figure 5. $M \times \tau(\phi)$ as the minimal model of ϕ with respect to $\leq_M$.

□

With this result in our disposal we can aim to directly construct an algorithm and a proof for the stopperedness of $\leq_M$ over $*_{\mathrm{FTSF}}$. The idea is that, since fair simulation preserves ACTL, we can use the product operation to produce a model that refines a designated model M, and also refines another model that satisfies a formula ϕ. Then, the product would have to satisfy ϕ by construction. This last model must, essentially, contain exactly all the possible behaviours allowed by the formula ϕ, in other words, to be a tableau of that formula. Several researchers have investigated such constructions for ACTL, e.g. [GL94, CGL96, Mai00, PMT02] with the following general properties:

Lemma 17. A function $\tau : \mathcal{L}_{\mathrm{ACTL}} \to \mathcal{M}_{\mathrm{FTSF}}$ can be defined such that:

for all $M \in \mathcal{M}_{\mathrm{FTSF}}, \tau(\phi) \leftarrow M$ iff $M \models \phi$.

Moreover, τ is computable and $|\tau(\phi)| \leq 2^{|\phi|}$. A corollary of the above is that

$\phi \models \psi \quad \text{iff} \quad \tau(\phi) \models \psi$.

The rest of the results in this section make use of tableau constructions, but only by making reference to their general properties mentioned above. A specific tableau construction, the one described in [CGL96], will be implemented and used in Section 5.3. We can now state the main result of this section. The following theorem forms the basis of the results concerning stopperedness and decidability for minimal refinement in ACTL.

Theorem 18. For any model N inside $M *_{\mathrm{FTSF}} \phi$, $N \leftrightarrows \tau(\phi) \times M$.

Proof. We prove that among the models of ϕ that are simulated by M, $\tau(\phi) \times M$ is the minimum, modulo fair simulation. Let N be a model of ϕ.

Assume that $M \not\leftarrow N$. In that case it follows trivially by the second clause of the definition of $\leq_M$ that $\tau(\phi) \times M \leq_M N$, since $M \leftarrow \tau(\phi) \times M$ as mentioned previously.

Thus, we assume that $M \leftarrow N$ (see Figure 5). Since N is a model of ϕ, it must be the case that $\tau(\phi) \leftarrow N$, because $\tau(\phi)$ is the tableau of ϕ. Therefore, $\tau(\phi) \times M \leftarrow N$ by applying Lemma 16. Also, it holds that $M \leftarrow \tau(\phi) \times M$, thus $\tau(\phi) \times M \leq_M N$.

Therefore, for all $N \in \mathrm{mod}_{\mathrm{FTSF}}(\phi)$, it holds that $\tau(\phi) \times M \leq_M N$. From the preservation of ACTL by fair simulation and the fact that $\tau(\phi) \leftarrow \tau(\phi) \times M$ it follows that $\tau(\phi) \times M$ is a model of ϕ and therefore is the minimum with respect to $\leq_M$ in $\mathrm{mod}_{\mathrm{FTSF}}(\phi)$. □

The above result answers the questions of stopperedness and decidability in one go. Specifically, the ordering $\leq_M$ is obviously stoppered over

$\mathcal{C}_{\mathrm{FTSF}} = \{\mathrm{mod}_{\mathrm{FTSF}}(\phi) \mid \phi \in \mathcal{L}_{\mathrm{ACTL}}\}$.

Moreover, since $\tau(\phi) \times M$ is finite and computable it follows that there is an effective procedure for checking $N \in M *_{\mathrm{FTSF}} \phi$, i.e. checking whether $N \leftrightarrows \tau(\phi) \times M$. Finally, for any $\psi \in \mathcal{L}_{\mathrm{ACTL}}$ there is an effective procedure for checking $M *_{\mathrm{FTSF}} \phi \models \psi$, i.e. checking whether $\tau(\phi) \times M \models \psi$ (observe that since ACTL is preserved by fair simulation there is no need to restrict ψ as was the case with modal logic).

Finally, we present a simple result that will be used later, in Section 5.3. Essentially, it states that the tableau of a conjunction can be viewed as the product of the tableaus of the conjuncts. This, in turn, characterises iteration over minimal refinements in ACTL: a sequence of minimal refinements is equivalent to a single minimal refinement by the conjunction of the formulae appearing in the sequence.

Lemma 19. For all $\phi, \psi \in \mathcal{L}_{\mathrm{ACTL}}$, $\tau(\phi \wedge \psi) \leftrightarrows \tau(\phi) \times \tau(\psi)$. Therefore, $M *_{\mathrm{FTSF}} \phi *_{\mathrm{FTSF}} \psi \leftrightarrows M *_{\mathrm{FTSF}} \phi \wedge \psi$.

Proof. By the properties of the product, it follows that $\tau(\phi) \leftarrow \tau(\phi) \times \tau(\psi)$. Fair simulation preserves ACTL, thus, since $\tau(\phi) \models \phi$ it follows that $\tau(\phi) \times \tau(\psi) \models \phi$ and similarly $\tau(\phi) \times \tau(\psi) \models \psi$ therefore $\tau(\phi) \times \tau(\psi) \models \phi \wedge \psi$. But then, by Lemma 17 we get that $\tau(\phi \wedge \psi) \leftarrow \tau(\phi) \times \tau(\psi)$.

Obviously $\tau(\phi \wedge \psi) \models \phi \wedge \psi$ therefore $\tau(\phi \wedge \psi) \models \phi$. By applying again Lemma 17 we obtain $\tau(\phi) \leftarrow \tau(\phi \wedge \psi)$ and similarly $\tau(\psi) \leftarrow \tau(\phi \wedge \psi)$. With the help of Lemma 16 we are led to $\tau(\phi) \times \tau(\psi) \leftarrow \tau(\phi \wedge \psi)$. This completes the proof. □

5.3. Implementation and case study

We have proceeded to develop a prototype implementation of minimal refinement for ACTL and FTSF. It is easy to see that the product operation is identical to the synchronous composition operation found in branching-time model checkers like SMV [McM93]. Moreover, since SMV provides an already existing framework for CTL (and thus ACTL) model-checking we decided to use it for the purpose of (implicitly) computing the product structure $\tau(\phi) \times M$, after having generated $\tau(\phi)$ with our implementation.

Therefore, our implementation must be able to produce the tableau model, in SMV code form. This code will later be combined with the SMV code describing the model the user wants to minimally refine, and fed to the SMV model checker. We chose to implement the tableau construction described in [CGL96], because it is an adequately powerful construct (in the sense that it serves as a tableau for the full language) while being relatively simple (it does not attempt to incorporate complex formalisms for the fairness constraints, but only sets of sets of states). We will review the construction here.

The basic idea is to decompose the given formula ϕ in its subformulae and build a model the states of which are appropriate subsets of those (slightly altered) subformulae.

Definition 16. The set $\mathrm{sub}(\phi)$ of the subformulae of ϕ and the set $\mathrm{el}(\phi)$ of elementary formulae of ϕ are defined as follows.

1. (a) If $\phi = \top$ or $\phi = \bot$, then $\mathrm{sub}(\phi) = \{\phi\}$.
   (b) If $\phi = p$ or $\phi = \neg p$ then $\mathrm{sub}(\phi) = \{\phi, p\}$.
   (c) If $\phi = \mathrm{AX}\psi$, then $\mathrm{sub}(\phi) = \{\phi\} \cup \mathrm{sub}(\psi)$.
   (d) If ϕ is of the form $\phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$ or $\mathrm{A}(\phi_1 \mathrm{U} \phi_2)$ or $\mathrm{A}(\phi_1 \mathrm{R} \phi_2)$, then $\mathrm{sub}(\phi) = \{\phi\} \cup \mathrm{sub}(\phi_1) \cup \mathrm{sub}(\phi_2)$.
2. (a) If $\phi = \top$ or $\phi = \bot$, then $\mathrm{el}(\phi) = \emptyset$.
   (b) If $\phi = p$ or $\phi = \neg p$ then $\mathrm{el}(\phi) = \{p\}$.
   (c) If $\phi = \phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$, then $\mathrm{el}(\phi) = \mathrm{el}(\phi_1) \cup \mathrm{el}(\phi_2)$.
   (d) If $\phi = \mathrm{AX}\psi$, then $\mathrm{el}(\phi) = \{\phi\} \cup \mathrm{el}(\psi)$.
   (e) If $\phi = \mathrm{A}(\phi_1 \mathrm{U} \phi_2)$ or $\phi = \mathrm{A}(\phi_1 \mathrm{R} \phi_2)$, then $\mathrm{el}(\phi) = \{\mathrm{AX}\bot, \mathrm{AX}\phi\} \cup \mathrm{el}(\phi_1) \cup \mathrm{el}(\phi_2)$.

where p is a propositional letter. The formula $\mathrm{AX}\bot$ denotes the lack of fair paths beginning at the state which satisfies this formula.

The tableau $\tau(\phi)$ of a formula ϕ is the ACTL model $\langle W, S, \mathcal{A}, v, \rightarrow, \mathcal{F} \rangle$ whose components are defined as follows. The state space W of $\tau(\phi)$ is the set $2^{\mathrm{el}(\phi)}$. Before we define the set of initial states and the transition relation, we will define a map sat from $\mathrm{el}(\phi) \cup \mathrm{sub}(\phi) \cup \{\top, \bot\}$ to subsets of $2^{\mathrm{el}(\phi)}$. Intuitively, $\mathrm{sat}(\psi)$ will be the set of states of the tableau that satisfy ψ.

1. $\mathrm{sat}(\psi) = \{\psi \in s\}$ where $\psi \in \mathrm{el}(\phi)$.

2. $\text{sat}(\neg p) = \{s \mid p \notin s\}$ where p is an atomic proposition.
3. $\text{sat}(\phi_1 \vee \phi_2) = \text{sat}(\phi_1) \cup \text{sat}(\phi_2)$.
4. $\text{sat}(\phi_1 \wedge \phi_2) = \text{sat}(\phi_1) \cap \text{sat}(\phi_2)$.
5. $\text{sat}(\text{A}(\phi_1 \text{U} \phi_2)) = (\text{sat}(\phi_2) \cup (\text{sat}(\phi_1) \cap \text{sat}(\text{AXA}(\phi_1 \text{U} \phi_2)))) \cup \text{sat}(\text{AX}\bot)$.
6. $\text{sat}(\text{A}(\phi_1 \text{R} \phi_2)) = (\text{sat}(\phi_2) \cap (\text{sat}(\phi_1) \cup \text{sat}(\text{AXA}(\phi_1 \text{R} \phi_2)))) \cup \text{sat}(\text{AX}\bot)$.

We can now define the missing pieces from the tableau construction.

Definition 17. The tableau $\tau(\phi) = \langle W, S, \mathcal{A}, v, \rightarrow, \mathcal{F} \rangle$ of a formula ϕ is defined as follows.

- $W = 2^{\text{el}(\phi)}$.
- $S = \text{sat}(\phi)$.
- $\mathcal{A} = \{p \mid p \in \text{el}(\phi)\}$.
- $v(s) = \{p \mid p \in s\}$.
- $s \rightarrow t$ if and only if $\bigwedge_{\text{AX}\psi \in \text{el}(\phi)} s \in \text{sat}(\text{AX}\psi) \Rightarrow t \in \text{sat}(\psi)$.
- $\mathcal{F} = \{(W \setminus \text{sat}(\text{AXA}(\phi_1 \text{U} \phi_2))) \cup \text{sat}(\phi_2) \mid \text{AXA}(\phi_1 \text{U} \phi_2) \in \text{el}(\phi)\}$.

The proof that this structure is indeed a tableau of ϕ in the sense of Lemma 17 can be found in [CGL96].

Our implementation consists of a set of classes in C++ that can be used for expressing an arbitrary ACTL formula and generating its tableau model, via Definition 17. Pictures of the model (in the DOT format [NDG+]) can be be produced for visualisation purposes. Moreover, code can be produced that describes the tableau in the SMV format. This implementation can be obtained from `http://www.cs.bham.ac.uk/~mdr/`.

This set of classes comprises of:

- A base class `Formula` and its derived classes, for the representation of ACTL formulae: `Conjunction`, `Disjunction` (propositional connectives), `AtomicFormula`, `NegatedAtomicFormula` (propositional variables and their negations), `Until` (AU), `Release` (AR) and `Next` (AX). These classes provide auxiliary methods, including methods for extracting the set of elementary formulae and subformulae of an instance.
- A class `Model` that handles all issues related to the building, visualising and converting the model to SMV code.
- A class `Tableau` that is derived from `Model`. This class contains the mechanisms for the building of the tableau model, given a `Formula`-type object that represents an ACTL formula.

The algorithm for producing the tableau is the direct one, i.e. symbolic techniques have not been used. This limits the usefulness of our implementation because the tableau reviewed above (and all the other tableaus from the literature that have been mentioned previously) is of exponential size in the length of the formula. Therefore, our implementation will have an exponential complexity in the length of the formula, unless significant progress is made in the symbolic techniques for producing these tableaus, or new tableau constructions that do not suffer from the state explosion problem are proposed. We describe possible avenues of research on this issue in Section 6.

In order to test the implementation and, more importantly, to examine the process from a practical perspective, a case study was performed on the well-known example of *mutual exclusion*. The purpose of this case study is to demonstrate that the need for minimal refinement arises naturally when designing a system, and that minimal refinement can automatically produce models that satisfy the given requirements and are as close as possible to the original model, in terms of refinement.

We chose the mutual exclusion problem because it is a simple, yet non-trivial example of a model/protocol, and because of its ubiquity in textbooks, e.g. in the book [HR04] (pages 187-199; or 181–200 in the first edition). The account appearing in this book is incremental, and the stages shown demonstrate the process of designing a model of a protocol which adheres to an increasing subset of the final specifications.

The problem of mutual exclusion arises in concurrent computation when shared resources are used. In particular, assume that two processes p_1 and p_2 are running on the same computer system by means of some kind of time-sharing (we will only consider two processes for simplicity). Assume further that, at times, p_1 and p_2 need to use some resource that cannot offer concurrent access, e.g. a printer. It is customary that the system employs some method of mutual exclusion that will ensure that the resource is not accessed simultaneously by the two processes. This can be accomplished through many techniques, either preemptive, such as an operating system-level scheduler, or cooperative, such as the use of semaphores. By *critical section* we will denote the duration within which a process is accessing the resource.

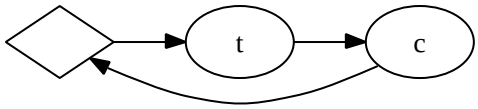

Figure 6. A model of a process.

Regardless of the specific techniques used, the scheduling must have a number of properties. The following four are discussed in [HR04]:

Safety: No two processes should ever be in their critical sections at the same time.

Liveness: If a process requests to enter its critical section, it will eventually be allowed to do so.

Non-blocking: A process can always request to enter its critical section.

No strict sequencing: There is no fixed order in which processes must enter their critical section.

We will model the processes from the point of view of the protocol used to enforce mutual exclusion. The processes, then, can be assumed to have three potential states of execution:

- A process can be doing work which is unrelated to the shared resource.
- Or, it may request to enter its critical section.
- Or, it may be executing inside its critical section.

Therefore we will use two propositional variables per process, t_i denoting that process i requests to enter its critical section, and c_i to mean that process i is inside its critical section. Figure 6 presents a transition system for a process. We are using an over-simplified model of a process, in the sense that each process has to change state in every clock tick, i.e. it is not allowed to stay in the same state. Also note that the initial state of the process is denoted by a diamond.

Two of these processes were modelled in SMV code and combined by asynchronous composition in SMV (note that this is not the same as the synchronous product construction). This means that at any given moment one process is chosen non-deterministically and is allowed to execute for one step. The state variables are now named t_1, t_2, c_1, c_2. The result is shown in Figure 7. The solid edges indicate that process 1 is executing and the dashed edges respectively for process 2. It is normally required that the scheduling is fair in that both processes are selected for execution infinitely often. This means that the set of solid edges has to be visited infinitely often by a computation path, and similarly for the set of dashed edges. Notice that the transition model actually constructed by the model checker is more complicated as the asynchronous composition operation is implemented by appropriate insertion of auxiliary variables. The distinction will be suppressed in the following, for reasons of readability of the produced models.

In this example, the constraints laid down earlier can be expressed formally.

Safety: $\mathrm{AG}(\neg c_1 \vee \neg c_2)$.

Liveness: $\mathrm{AG}(t_i \rightarrow \mathrm{AF}(c_i))$.

Non-blocking: $\mathrm{AG}(\neg t_i \wedge \neg c_i \rightarrow \mathrm{EX}t_i)$.

No strict sequencing: $\mathrm{EG}(c_1 \wedge \mathrm{E}(c_1 \mathrm{U}(\neg c_1 \wedge \mathrm{E}(\neg c_2 \mathrm{U} c_1))))$.

Obviously, the non-blocking and non strict sequencing properties cannot be expressed by ACTL formulae, so we will not deal with them. Moreover, they happen to be satisfied by the models we will consider.

A further fairness constraint usually employed in this case is that we do not consider computation paths that spend an inordinate amount of time in a process' critical section. These constraints are expressed by the formulae $\neg c_1$ and $\neg c_2$. In other words, we force the system to only consider paths that visit an infinite number of times the non-critical parts of the processes.

Obviously, the state $c_1 c_2$ has to be excluded from the set of reachable states. We basically want to impose the formula $\phi = \mathrm{AG}(\neg c_1 \vee \neg c_2)$, an ACTL formula, to the above model. But we want to retain the computation paths that are, in a sense, independent from ϕ. A first test for the minimal refinement operation is to produce $M_1 *_{\mathrm{FTSF}} \phi$.

Concerning the tableau of the safety formula $\phi = \mathrm{AG}(\neg c_1 \vee \neg c_2)$, observe that $\phi = \mathrm{A}(\bot \mathrm{R}(\neg c_1 \vee \neg c_2))$.

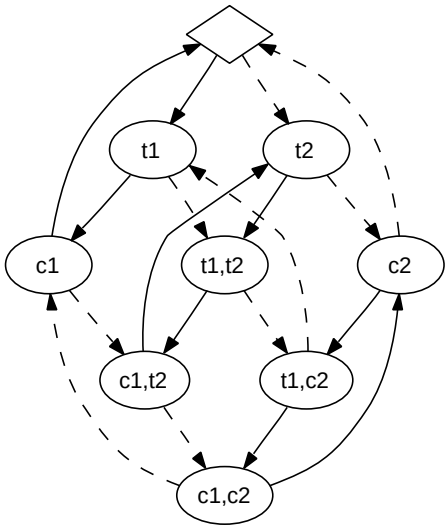

Figure 7. M_1, the asynchronous composition of two processes.

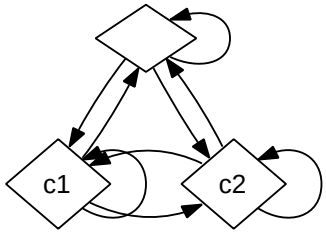

Figure 8. The tableau of $\mathrm{AG}(\neg c_1 \vee \neg c_2)$.

Therefore, $\mathrm{el}(\phi) = \{\mathrm{AX}\bot, \mathrm{AX}\phi, c_1, c_2\}$, so the tableau can have up to 16 states. However, we may remove the states that are unreachable from the initial ones, as well as the states that have no successors. The resulting model can be seen in Figure 8. Variables introduced by the tableau construction that have the same truth value across all reachable states have been suppressed. Also, note that there are no fairness constraints.

Because of the simplicity of the derived tableau, the product structure M_2 of M_1 and $\tau(\mathrm{AG}(\neg c_1 \vee \neg c_2))$ is the expected, i.e. M_1 without the state c_1c_2 (figure 9). The fairness constraints remain the same, i.e. the sets of states with $\neg c_1$ and with $\neg c_2$. Model M_2 is identical to the model that appears as the 'first attempt' for the mutual exclusion protocol in [HR04], p.188 (p.170 in the first edition).

We observe that although the safety, non-blocking and no strict sequencing properties are true of M_2, liveness fails. This is because there exists a computation path which goes through the states $t_1 \to t_1, t_2 \to t_1, c_2$ and then loops back to t_1, ad infinitum. This path violates the liveness property because even though process 1 is requesting to enter its critical section, it never manages to do so. We will attempt to repair this by minimally refining M_2 with the liveness property. The liveness property is

$$\mathrm{liveness}_i = \mathrm{AG}(t_i \to \mathrm{AF}c_i) = \mathrm{AG}(\neg t_i \vee \mathrm{AF}c_i)$$

which is obviously an ACTL formula. The goal is to minimally refine by the formula $\mathrm{liveness}_1 \wedge \mathrm{liveness}_2$. However, this leads to a tableau that is unnecessarily large, so as to increase substantially verification times: notice that since $\mathrm{liveness}_1$ and $\mathrm{liveness}_2$ do not share any propositional variables, the structure $\tau(\mathrm{liveness}_1 \wedge \mathrm{liveness}_2)$ will have exactly as many states as the product of the number of states of the structures $\tau(\mathrm{liveness}_i)$.

Under the present framework this cannot be avoided. However, we can make use of Lemma 19 and perform two minimal refinements. The advantage is that SMV is quite efficient when dealing with product structures so we do not have to re-invent this efficiency for our algorithms. The model for $\tau(\mathrm{liveness}_1)$ is shown in Figure 10: note that since the model is nearly total, in the sense that every state is connected to almost

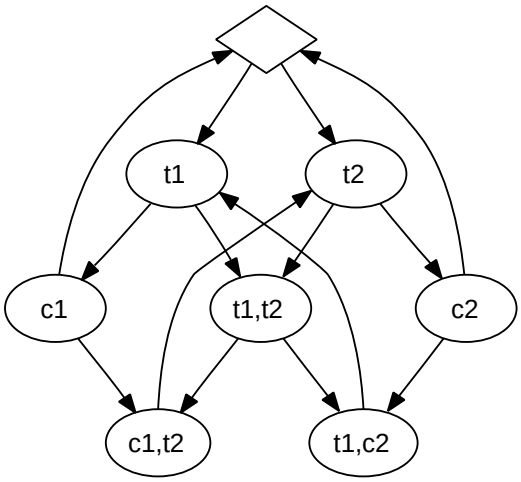

Figure 9. M_2, the composition of M_1 and $\tau(\mathrm{AG}(\neg c_1 \vee \neg c_2))$.

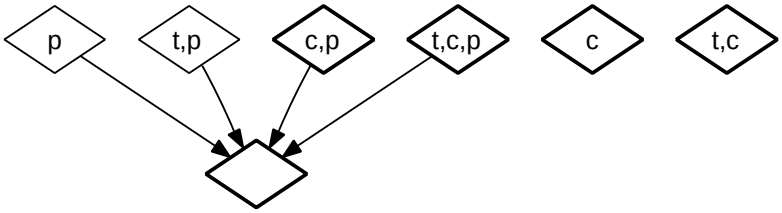

Figure 10. $\tau(\text{liveness}_1)$, with edges shown wherever they *do not* exist in the tableau.

every other, the edges shown in the figure are the ones that *do not exist* in the actual tableau. Notice that in this tableau, the insertion of auxiliary variables cannot be ignored because one of them $p = \mathrm{AXAF}c_1$ varies in truth value across the reachable states. Moreover, this tableau imposes a non-trivial fairness constraint:

$$(\neg t \wedge \neg p) \vee (t \wedge \neg p \wedge c) \vee (p \wedge c).$$

The states that satisfy this condition are drawn bold in Figure 10. These must be visited infinitely often by the considered computation paths. The introduction of fairness constraints is due to the need to satisfy eventualities that derive from the *until* operator, implicit in AF.

Finally, C, the composition of M_2, $\tau(\text{liveness}_1)$ and $\tau(\text{liveness}_2)$ is shown in Figure 11. Another, perhaps more readable version of C is shown in Figure 12. Boxes are drawn around states that have identical observable valuations, i.e., in terms of t_i and c_i. Transitions that end at these boxes are meant as abbreviations of a set of transitions leading to each state in the box.

It is easy to see that the loop $t_1 \to t_1, t_2 \to t_1, c_2$ and back to t_1 still exists, as it should since any finite number of iterations of this loop should be permitted. However, the fairness constraints are now non-trivial because the following ones have been added through the composition with the liveness tableaus:

$$(\neg t_i \wedge \neg p_i) \vee (p_i \wedge c_i)$$

for $i = 1, 2$. This formula is derived from the previously mentioned fairness constraint introduced by $\tau(\text{liveness}_i)$, plus the observation that states that satisfy t_i and c_i do not exist. Given these constraints, it is now impossible to execute the loop $t_1 \to t_1, t_2 \to t_1, c_2$ for an infinite number of times, since such a path would not visit infinitely often (actually, never) a state that satisfies $(\neg t_1 \wedge \neg p_1) \vee (p_1 \wedge c_1)$. Indeed, when model checked by SMV, the model C was found to satisfy the liveness formulae.

The model C' appearing in [HR04] as the second attempt for the mutual exclusion protocol is shown in Figure 13. As C, it satisfies the safety and liveness properties. Also, it obviously differs from C and is not equivalent to it, neither in terms of fair bisimulation or simulation. However, it can be shown that $C \leftarrow C'$. That is to say, C is strictly closer to the original model M_2 in terms of fair simulation, than C'

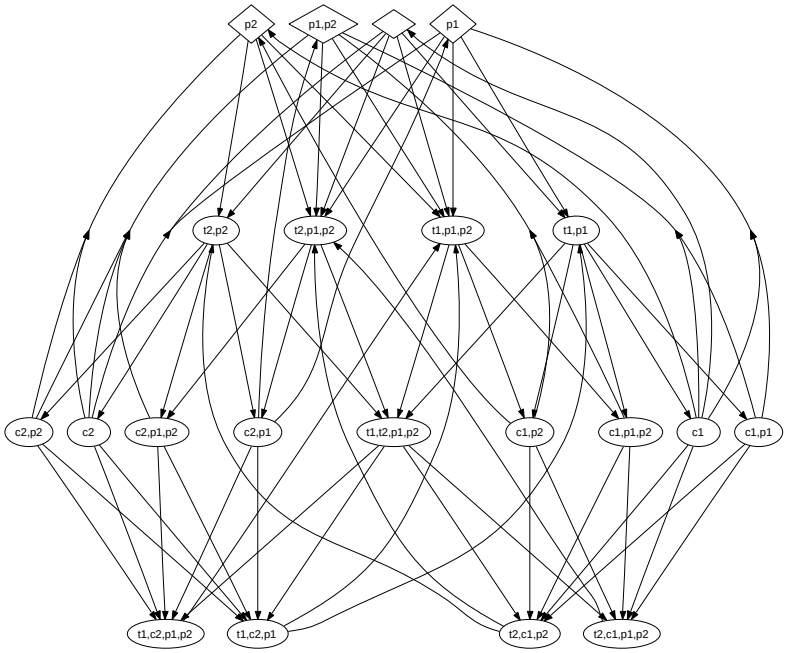

Figure 11. C, the composition of M_2, $\tau(\text{liveness}_1)$ and $\tau(\text{liveness}_2)$.

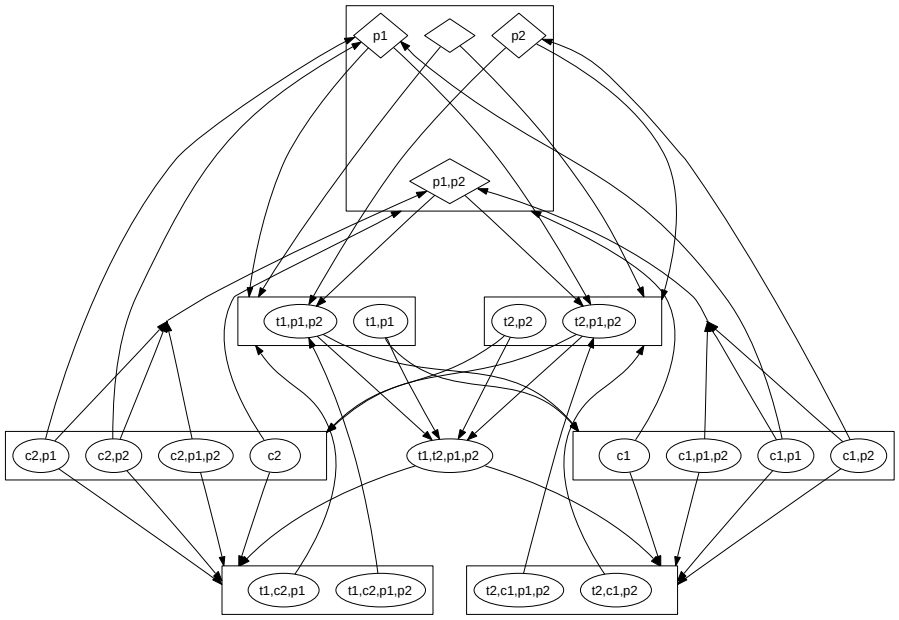

Figure 12. C, reformatted for readability.

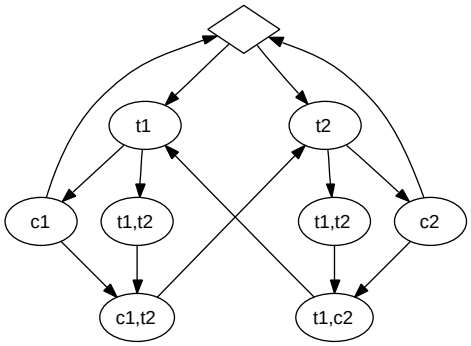

Figure 13. C', the "second attempt" for the mutual exclusion protocol.

is. This property, which is to be expected due to the definition of minimal refinement, reflects the fact that in producing C from M_2 no behaviours of M_2 that are compatible with the liveness properties have been removed. On the contrary, C' exhibits a strict temporal behaviour: if process 1 requests to enter its critical section when process 2 is not (state t_1), then it is guaranteed that process 1 will enter its critical section in at most 2 time steps.

This concludes the case study. What has been shown, we believe, is that minimal refinement arises naturally in the process of designing algorithms or models. Moreover, it seems that when it does arise, solutions constructed by hand are sometimes inferior in that they may introduce ad hoc restrictions that do not necessarily follow by the new requirements added to a model. On the other hand, minimal refinement may produce models that are very large in terms of state spaces, and that are difficult to understand.

6. Conclusions and further work

The conclusions, unresolved problems and open questions encountered are presented in this section.

Minimal refinement, described in Sections 3, 4 and 5.2, is a method for changing a given model so that the result refines it minimally while satisfying a new requirement. Using this method, a designer can obtain a revised design out of an old one and a new requirement. Moreover, the relationship between the resulting model and the initial one is well-defined: all the behaviours the newly obtained one exhibits are behaviours allowed by the initial system, and, moreover, the set of behaviours is maximal in that it is the largest one that satisfies the new property.

Minimal refinement has been studied under three frameworks:

- In modal logic, we studied minimal refinement over the class of m-saturated models with sets of sentences as the requirements (local and global satisfaction).
- Again, in modal logic, minimal refinement over finite structures was studied, with modal formulae as requirements (local and global satisfaction).
- Finally, we studied minimal refinement over transition systems with fairness constraints with formulae of ACTL as requirements.

In all of these cases, the conditions under which the minimal refinement is non-trivial were characterised (Lemmas 4, 9 and 14). In the temporal case as well as in the finite modal one, non-triviality was proved to be decidable (in the same lemmas). The ordering $\leq_M$ was proved to be stoppered (Theorems 6, 10 and 18). Lastly, effective ways to represent minimal refinements in the finite modal case and the temporal one, were presented in results 11, 12, 13 and 18.

Finally, a sample implementation of minimal refinement in the case of ACTL was presented in Section 5.3, along with a detailed study of an example of a mutual exclusion protocol. Using this implementation, a designer can provide an ACTL formula ϕ and a system M described in the language of the SMV model checker; the implementation will then produce the minimal refinement $M *_{\mathrm{FTSF}} \phi$, coded in the SMV language. The result can then be fed into the SMV model checker where the designer can verify the result against any further requirements.

Many possibilities exist for further work, as many questions are open with respect to minimal refinement. For the case of modal logic and finite structures, an obvious research goal is to find tractable algorithms for constructing minimal models. For the moment, the obvious and naive algorithms for the modal case of minimal refinement are of non-deterministic exponential complexity.

Studying the computational complexity of the associated problems is another obvious research avenue. It is easy to see that these problems will have an expensive worst-case complexity. After all, modal satisfiability is PSPACE-complete, regardless of bounds on the number of propositional atoms or modalities in the language [Hal95], and the test for non-triviality can easily be modified so as to solve modal satisfiability, rendering it PSPACE-hard in the length of the formula. On the other hand, it is also easy to show that its complexity is linear in the size of the model provided.

In conjunction to this, additional work can be focused on characterising more precisely the structure of the set of the resulting models, since we have obtained only very pessimistic upper bounds on its size.

In the case of temporal logics, it would be desirable to extend the set of results to languages that are not universally quantified (like ACTL) and therefore are not preserved by refinement. Unfortunately, as the results we have obtained on ACTL depend crucially on this property (and more specifically, the tight relationship between refinement, language inclusion and satisfaction), they are not transferable to such new languages and therefore a whole new approach would need to be developed.

Another direction for future work is the investigation of the tractability of the method in the case of ACTL. We are currently using the tableau construction introduced in [CGL96], which is of exponential size in the length of the formula. Thus, the minimal refinement may be of intractable size, as it is the product of the initial model and the tableau. One improvement that comes at the cost of some expressiveness is to use safety-ACTL[1], for which there is a more compact tableau [KGG99], although still exponential in the worst case.

Finally, further research could focus on ways to address this state-explosion problem from its implementation aspect, either by considering improvements on the tableau construction or by looking into the application of symbolic methods for generation of $M *_{\mathrm{FTSF}} \phi$.

A. A version of Zorn's lemma appropriate for stopperedness

The version of this lemma usually found in textbooks is the following:

Lemma 20. If every non-empty chain of a non-empty partially ordered set X has a lower bound in X, then X possesses a minimal element.

In this paper we use the following, slightly modified, version of Zorn's lemma as it is closer to the notion of stopperedness:

Lemma 21. If every non-empty chain of a non-empty set X equipped with a *preorder* $\leq$ has a lower bound in X, then for any element $x \in X$ there exists an element $y \in X$ such that $y \leq x$ and y is $\leq$-minimal in X.

Proof. Firstly we define a partial order $\sqsubseteq$ on the basis of $\leq$ as $x \sqsubseteq y$ iff $x < y$ or $x = y$, where $<$ is the strict counterpart of $\leq$ and x, y are members of X. This definition makes $\sqsubseteq$ a partial order as it is transitive, reflexive and antisymmetric.

Assume $x \in X$ and define $X' = \{x' \in X \mid x' \sqsubseteq x\}$. Obviously, X' is non-empty and it is also partially-ordered, as a restriction of a partial order remains a partial order.

Every non-empty chain C of X' with respect to $\sqsubseteq$ is also a non-empty chain of X. Therefore, by assumption, it has a lower bound c within X. But since $C \subseteq X'$, it follows that $c \sqsubseteq x$ and therefore $c \in X'$. The conditions of the original version of Zorn's lemma are satisfied, thus, X' has a minimal element y, with respect to $\sqsubseteq$. Since by its definition, X' is downward-closed, y is minimal in X too.

From y's minimality with respect to $\sqsubseteq$ it follows that there is no $z \in X'$ such that $z \sqsubset y$. Equivalently, through the definition of $\sqsubseteq$ and some algebra, $z < y \wedge y \not< z \wedge y \neq z$. This, in turn, by the necessary antisymmetry and anti-reflexivity of a strict counterpart of an ordering, is equivalent to $z < y$. Therefore, y is minimal with respect to $\leq$ in X'. This completes the proof. □

[1] Safety-ACTL is the fragment of ACTL obtained by only using AX and AW (the weak counterpart of *until*).

Acknowledgments. We are grateful for the thorough reviews and constructive suggestions made by the anonymous reviewers.

References

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., 1990. IEEE Computer Society Press.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[BEF93] J. van Benthem, J. van Eijck, and A. Frolova. Changing preferences. Technical Report CS-93-10, Centre for Mathematics and Computer Science, Amsterdam, 1993.

[BFG+91] A. Bouajjani, J. C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for branching time semantics. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, ICALP'91*, volume 510 of *LNCS*, pages 76–92, Madrid, Spain, July 1991. Springer.

[BMP97] H. Bezzazi, D. Makinson, and R. P. Pérez. Beyond rational monotony: Some strong non-horn rules for nonmonotonic inference relations. *Journal of Logic and Computation*, 7(5):605–631, 1997.

[CE81] E. M. Clarke and E. A. Emerson. The design and synthesis of synchronization skeletons using temporal logic. In *Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–72, Yorktown Heights, New York, 1981. Springer-Verlag.

[CES86] 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.

[CGL96] E. M. Clarke, O. Grumberg, and D. Long. Model checking. Nato ASI Series F, volume 152, Springer-Verlag, 1996. Marktoberdorf summer school.

[Che80] B.F. Chellas. *Modal Logic: an introduction*. Cambridge University Press, 1980.

[Dal88] M. Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 475–479, St. Paul, Minnesota, August 1988.

[dR95] M. de Rijke. Modal model theory. Technical Report CS–R9517, CWI, Amsterdam, 1995.

[GL94] O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, May 1994.

[Gor03] N. Gorogiannis. *Computing Minimal Changes of Models of Systems*. PhD thesis, School of Computer Science, University of Birmingham, June 2003.

[GR02] N. Gorogiannis and M. D. Ryan. Requirements, specifications and minimal refinement. In *9th Workshop on Logic, Language, Information and Computation*, volume 67 of *Electronic Notes in Theoretical Computer Science*, Brazil, September 2002. Elsevier.

[Gro88] A. Grove. Two modelings for theory change. *Journal of Philosophical Logic*, 17:157–170, 1988.

[Hal95] J. Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372, 1995.

[HKR97] T. A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. In *Proceedings of the Ninth International Conference on Concurrency Theory (CONCUR)*, volume 1243 of *Lecture Notes in Computer Science*, pages 273–287. Springer-Verlag, 1997.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[Hol95] M. Hollenberg. Hennessy-Milner classes and process algebra. In A. Ponse, M. de Rijke, and Y. Venema, editors, *Modal Logic and Process Algebra*, pages 187–216. CSLI Publications, 1995.

[HR04] M. R. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, second edition, 2004.

[KGG99] S. Katz, O. Grumberg, and D. Geist. "Have I written enough properties?" - A method of comparison between specification and implementation. In *Conference on Correct Hardware Design and Verification Methods*, pages 280–297, 1999.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[KM89] H. Katsuno and A. O. Mendelzon. A unified view of propositional knowledge base updates. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1413–1419, Detroit, Michigan, USA, August 1989.

[KM91] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[KM92] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In P. Gärdenfors, editor, *Belief Revision*, Cambridge Computer Tracts, pages 183–203. Cambridge University Press, Cambridge, 1992.

[Mai00] M. Maidl. The common fragment of CTL and LTL. In *Proceedings of the 41th Annual Symposium on Foundations of Computer Science*, pages 643–652, 2000.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[Mey91] R. van der Meyden. A clausal logic for deontic action specification. In *Proceedings of the International Logic Programming Symposium*, pages 221–238, San Diego, October 1991. MIT Press.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489, London, UK, September 1971.

[NDG+] S. North, D. Dobkin, E. Gansner, E. Koutsofios, K. Vo, and G. Woodhull. Graphviz, a suite of tools for visualizing graphs. Can be obtained from `http://www.graphviz.org`.

[PMT02] H. Peng, Y. Mokhtari, and S. Tahar. Environment synthesis for compositional model checking. In *Proceedings of the IEEE International Conference on Computer Design*, pages 70–75. IEEE Computer Society Press, September 2002.