Weak preservation of CTL* and its sublogics

Yannis Kassios

February 19, 2002

Abstract

This survey describes the subject of weak preservation in CTL*. Weak preservation is usually studied in the context of abstraction. However, we first describe sufficient conditions for weak preservation in the concrete state space. We then independently examine what information for the concrete system is preserved when model checking various abstract versions. These two different results are combined in the end, to explain how full CTL* is weakly preserved in abstract model checking. This presentation simplifies and gives insight on the workings of Dennis Dam's abstraction framework, which seems to be the most important current contribution to the topic.

1 Introduction

In most interesting cases of model checking, we cannot cope with the problem of state explosion without some sort of abstraction. Abstraction is a mechanism that throws away information, which is necessary if model checking is to preserve its practicality. On the other hand the lost information may be important to the question in hand, so we have to make sure that there is a mathematically sound scheme out of which we can extract the correct information about the concrete system, while model checking in an abstract context.

Weak preservation is such a scheme: If our concrete question is C and the corresponding abstract question is A , weak preservation is the requirement that

$$A \Rightarrow C$$

If for an abstraction scheme this property is true, then we can deduct properties of the concrete system by model checking the abstract system. In theorem proving, this is equivalent to forward deduction.

Understandably, weak preservation is normally studied in the context of a specific abstraction (e.g. state abstraction [2], [6]). However in reality it is a concept separated from abstraction itself. The question of whether a property is weakly preserved can be asked in concrete transition systems too and the answer is simpler, since it doesn't involve any unnecessary complication related to abstraction considerations. In this paper we undertake this task: to study weak preservation and state abstraction as two different concepts in their own right and to demonstrate how the combination of those two different results gives us a currently known abstraction scheme [6]. Since the two problems are studied separately, the presentation is much simpler than that of the referenced papers.

The rest of the paper is as follows: Sections 2 and 3 contain the important basic definitions that will be necessary for the following sections. More specifically section 3 deals with the concept of the abstract state space. Some basic definitions slightly deviate from their standard counterparts in literature. Section 4 is a study of the monotonicity principles in CTL*, directly related to weak preservation in the concrete state space. Section 5 studies abstraction schemes that were introduced in [2] (for ∀CTL*) and [6] (for CTL*) and explains, based on the results of the previous section, why they suffice for the weak preservation problems they treat. This explanation is based on theorems that relate the results of abstract model checking with equivalent results of concrete model checking. Finally, section 6 summarizes the paper and concludes.

Notation: We use $\boldsymbol{=}$ and $\boldsymbol{\Rightarrow}$ as least-precedence versions of $=$ and $\Rightarrow$ respectively. The notation is due to Hehner [7].

2 Models and CTL*

2.1 Basic definitions

In the framework we are working in, there is a common infinite set of concrete states $\mathcal{S}$.

Definition 1 (Propositional formulas) *A* propositional formula *is a subset of $\mathcal{S}$. A propositional formula that contains only one state s is called the* state formula *of s .*

When using propositional formulas, we use notation that's different from set theory and closer to propositional calculus. Let p and q be propositional formulas

$$\begin{array}{ll} & true = \mathcal{S} \\ \wedge & false = \emptyset \\ \wedge & p \;\&\; q = p \cap q \\ \wedge & p \mid q = p \cup q \\ \wedge & \sim p = \mathcal{S} - p \\ \wedge & p \rightarrow q = \sim p \mid q \end{array}$$

We also use the $\bigcup$ and $\bigcap$ quantifiers to denote the application of $|$ and $\&$ respectively to many operands, for example:

$$\bigcap_{i \in \{1,2,3\}} p_i \;=\; p_1 \& p_2 \& p_3$$

Definition 2 (Transition system) *A* transition system *on $\mathcal{S}$ (or concrete transition system, or transition system), is a subset of $\mathcal{S}^2$. If T is a transition system and $s \in \mathcal{S}$ then $T(s)$ is the propositional formula that is defined by*

$T(s) = \bigcup_{t \in S \wedge (s,t) \in T} \{t\}$

Definition 3 (Path) *A* path *for a transition system T is a total function $\pi : \mathbb{N} \rightarrow \mathcal{S}$, such that:*

$\forall i \in \mathbb{N} \cdot (\pi(i+1) \in T(\pi(i)))$

The set of all paths π for T such that $\pi(0) = s$ is denoted $Path(T, s)$:

$Path(T, s) = \{\pi \in \mathbb{N} \rightarrow \mathcal{S} \mid \forall i \in \mathbb{N} \cdot (\pi(i+1) \in T(\pi(i)) \quad \wedge \quad \pi(0) = s\}$

Definition 4 (Reachability) *Given a transition system T and a state s , we say that a state t is* reachable *in T starting with s if*

$\exists i \in \mathbb{N}, \pi \in Path(T, s) \cdot (t = \pi(i))$

The set of all reachable states for T and s is denoted $Reach(T, s)$:

$Reach(T, s) = \{t \in \mathcal{S} \mid \exists i \in \mathbb{N}, \pi \in Path(T, s) \cdot (t = \pi(i))\}$

Definition 5 (Completeness) *Given a transition system T and a state s , we say that T is* complete *starting with s if*

$$\begin{array}{ll} & \forall t \in Reach(T, s) \cdot (T(t) \neq \emptyset) \\ \wedge & Path(T, s) \neq \emptyset \end{array}$$

Definition 6 (Model) *A* model *is a pair (s, T) where I is a state and T is a transition system such that:*

- *T is complete starting with s .*
- *$Reach(T, s)$ is finite.*

For a model $M = (s, T)$ we denote

$Reach(M) = Reach(T, s) \wedge Path(M) = Path(T, s)$

Our definition doesn't allow for non-determinism for the first state, since this makes the treatment of CTL* formulas more complicated. There's no loss of generality, since transition systems *are* non-deterministic. If we want for example to have a non-deterministic choice from initial states s_1 and s_2, we can designate one initial state s_0 with transitions to s_1 and s_2.

Other than the determinism in initial states, our definition of a "model" is analogous to Kripke structures often used in literature, but with less redundancy (propositional formulas are identified with the state sets they represent).

Let $M = (I, T)$ and $N = (I', T')$ be models. Model equivalence is weaker than $I = I' \wedge T = T'$, since we are not interested in non-reachable states:

Definition 7 (Model equivalence) *Two models M and N are* equivalent *and we write $M = N$ if and only if*

$$Path(M) = Path(N)$$

The condition $Path(M) = Path(N)$ is equivalent to:

$$Reach(M) = Reach(N) \;\wedge\; \forall s \in Reach(M) \cdot (T(s) = T'(s))$$

2.2 CTL*

Definition 8 (State and path formulas) *CTL* defines two sorts of formulas,* state *and* path formulas*:*

- *Any propositional formula is a state formula*
- *Any state formula is a path formula*
- *For a state formula ϕ, the formulas $\mathsf{X}\phi$, $\mathsf{G}\phi$ and $\mathsf{F}\phi$ are path formulas*
- *If ϕ and ϕ' are state formulas then $\phi \mathsf{U} \phi'$ is a path formula*
- *For a path formula ψ, the formulas $\forall\psi$ and $\exists\psi$ are state formulas*
- *If ϕ and ϕ' are state formulas then so are $\phi \& \phi'$ and $\phi \mid \phi'$*
- *If ψ and ψ' are path formulas then so are $\psi \& \psi'$ and $\psi \mid \psi'$*

Note that we don't allow negation in CTL* formulas except when it comes to the propositional formulas. This doesn't reduce expressiveness, as one can always use DeMorgan's laws to derive the 'canonical' form described above.

The $\models$ ("models") relation is defined recursively[1]:

[1] The following definition is consistent, even though it's ambiguous. For example, if p_1 and p_2 are propositional formulas, then $p_1 \mid p_2$ is a state formula both because it is a propositional formula and because it is the application of $\mid$ to two state formulas. The relation $\models$ is defined in two ways, but the values of $M, s \models p_1 \mid p_2$ given by the two definitions are equal

Definition 9 ("models" relation) *Let $M = (s, T)$ be a model, ϕ and ϕ' be state formulas, ψ and ψ' be path formulas, $s \in \mathcal{S}$, $\pi \in Path(T, \pi(0))$, p be a propositional formula. Then:*

$$
\begin{array}{ll}
 & (M \models \phi \; = (T, s) \models \phi) \\
\wedge & ((T, s) \models p \; = s \in p) \\
\wedge & ((T, s) \models (\phi \& \phi') \; = (T, s) \models \phi \; \wedge \; (T, s) \models \phi') \\
\wedge & ((T, s) \models (\phi \mid \phi') \; = (T, s) \models \phi \; \vee \; (T, s) \models \phi') \\
\wedge & ((T, s) \models (\forall \psi) \; = \forall \pi \in Path(T, s) \cdot ((T, \pi) \models \psi)) \\
\wedge & ((T, s) \models (\exists \psi) \; = \exists \pi \in Path(T, s) \cdot ((T, \pi) \models \psi)) \\
\wedge & ((T, \pi) \models \phi \; = (T, \pi(0)) \models \phi) \\
\wedge & ((T, \pi) \models \mathsf{X}\phi \; = (T, \pi(1)) \models \phi) \\
\wedge & ((T, \pi) \models \phi \mathsf{U} \phi' \; = \exists i \in \mathbb{N} \cdot ((T, \pi(i)) \models \phi' \; \wedge \; \forall j \in \cdot (j \leq i \Rightarrow (T, \pi(j)) \models \phi))) \\
\wedge & ((T, \pi) \models (\psi \& \psi') \; = (T, \pi) \models \psi \; \wedge \; (T, \pi) \models \psi') \\
\wedge & ((T, \pi) \models (\psi \mid \psi') \; = (T, \pi) \models \psi \; \vee \; (T, \pi) \models \psi')
\end{array}
$$

We write $X \not\models Y$ for any kind of expressions X and Y , to abbreviate $\neg(X \models Y)$.

Model equivalence respects the models relation in that for two equivalent models M and N and a state formula ϕ :

$$M \models \phi = N \models \phi$$

which is proved easily by induction.

Definition 10 (∀CTL* and ∃CTL*) *The sublogic ∀CTL* accepts only universal path quantifications. The sublogic ∃CTL* accepts only existential path unifications.*

3 Abstraction basics

3.1 Weak and strong preservation

Abstraction in model checking kicks in when we are unable to model check large transition systems. What we do instead is prove that model checking another simpler system will give a result that implies (or is equivalent to) the result that we're after, and then we model check that system. More formally, let M be a model and ϕ be a CTL* property. The problem of abstraction is viewed as choosing a simpler "abstract" model A such that, provably,

$$A \models \phi \Rightarrow M \models \phi$$

which is called *weak preservation*, or, more strongly,

$$A \models \phi = M \models \phi$$

which is called *strong preservation*.

In this project we concerns ourselves with weak preservation of CTL* properties.

3.2 Abstract states

An abstract state[2] is meant to be a collection of many concrete states, so technically it needs to be a non-singleton satisfiable propositional formula. We will disperse with the non-singleton requirement, since some abstractions may map a single concrete state to an abstract state:

Definition 11 (Abstract state) *An* abstract state *is a satisfiable propositional formula*

Definition 12 (Abstract state space) *An* abstract state space $\mathcal{A}$ *is an equivalence partition of* $\mathcal{S}$ *, that is,* $\mathcal{A}$ *contains abstract states, and for all* $a, b \in \mathcal{A}$ *it is:*

$$a = b \vee a \cap b = \emptyset$$

We may consider $\mathcal{A}$ to be our state space instead of $\mathcal{S}$ and model-check transition systems that are defined on this model. We will show later how our results translate to the "concrete" state space $\mathcal{S}$.

Given a CTL* formula ϕ_A on $\mathcal{A}$ we can get the corresponding CTL* formula for the concrete state by changing all propositional formulas with ϕ_A to their corresponding concrete propositional formulas:

Definition 13 (Corresponding abstract/concrete formulas) *Let* $\mathcal{S}$ *be a concrete state space and* $\mathcal{A}$ *an abstract state space on* $\mathcal{S}$ *. Let* ϕ_A *be a state CTL* formula on* $\mathcal{A}$ *and* ψ_A *be a path CTL* formula on* $\mathcal{A}$ *. The CTL* state formula* ϕ *and path formula* ψ *on* $\mathcal{A}$ *are the corresponding concrete formulas of* ϕ_A *and* ψ_A *respectively if we substitute any propositional subformula* p_A *in* ϕ_A *and* ψ_A *with*

$$p = \bigcup_{a \in p_A} a$$

For example, suppose that there are five concrete states $s_i, i \in \{1, .., 5\}$ and two abstract states $a = \{s_1, s_2\}$ and $b = \{s_3, s_4, s_5\}$. Then the abstract CTL* formula

$$\forall \mathsf{G}(a \rightarrow \mathsf{X} b)$$

corresponds to the concrete CTL* formula

[2]We will from now on deviate from the notation of [6] that depends largely on abstract interpretations [4] for its definition of abstract states and abstract transition systems

$\forall \mathsf{G}((s_1 \mid s_2) \rightarrow \mathsf{X}(s_3 \mid s_4 \mid s_5))$

Note that there is always a way to go from the abstract to the concrete formula, but not the other way around: some concrete formulas cannot be translated in some abstract spaces. In our example we cannot translate the concrete formula

$s_1 \mid s_3$

because our partition does not permit it. In general, given a partition $\mathcal{A}$ and a CTL* (state or path) concrete formula, the formula is translatable if any of its propositional subformulas p "respects the partition":

$\forall a \in \mathcal{A} \cdot (a \subseteq p \vee a \cap p = \emptyset)$

Definition 14 (Compatible formula) *A CTL* state or path formula, whose propositional subformulas satisfy the above requirement, will be called* compatible *with partition* $\mathcal{A}$ *.*

4 Weak preservation without abstraction

Before going on to use abstraction, we will inquire exactly how we achieve weak preservation in the concrete state space.

4.1 The lattice of models

There is a lattice structure on transition systems, whose ordering is $\subseteq$, the subset relation on $\mathcal{S}^2$. The top and bottom of this lattice are respectively $\mathcal{S}^2$ and $\emptyset$. Getting the image of a state is monotonic with respect to this lattice:

$T_1 \subseteq T_2 \Rightarrow \forall s \in \mathcal{S} \cdot (T_1(s) \subseteq T_2(s))$

The following relationship between paths and the model lattice is obvious:

Lemma 1 *For transition systems* T_1 *and* T_2 *and a state* s *:*

$T_1 \subseteq T_2 \Rightarrow Path(T_1, s) \subseteq Path(T_2, s) \wedge Reach(T_1, s) \subseteq Reach(T_2, s)$

Proof: Assume $T_1 \subseteq T_2$ and $\pi \in Path(T_1, s)$. Then for all $i \in \mathbb{N}$:

$\pi(i+1) \in T_1(\pi(i)) \subseteq T_2(\pi(i))$

which means $\pi \in Path(T_2, s)$ and therefore $Path(T_1, s) \subseteq Path(T_2, s)$. A similar argument goes for $Reach(T_1, s) \subseteq Reach(T_2, s)$. QED.

We will show now that *pure* formula model-checking is monotonic with respect to this lattice, where "pure formula" as follows:

Definition 15 (Pure formula) *A pure CTL^* state formula is of the form $\forall\psi$ or $\exists\psi$, where ψ doesn't contain path quantifications. We will call $\forall\psi$ a* universal *pure formula and $\exists\psi$ an* existential *pure formula. ψ will be called a* pure path formula*.*

Theorem 1 *Let ϕ be a pure formula and let $(T_1, s) \models \phi$. If ϕ is universal and $T_2 \subseteq T_1$, then $(T_2, s) \models \phi$. If ϕ is existential and $T_1 \subseteq T_2$, then $(T_2, s) \models \phi$.*

Proof: Let $\pi \in Path(T_1, s)$. Since ϕ is pure then it is of the form $Q\psi$, where Q is a path quantifier and ψ is a path formula without path quantifiers. ψ only contains propositional formulas and X and U operators, therefore, by definition of $\models$, the question $(T_1, \pi) \models \psi$ does not depend on T , as can be easily proved by structural induction on pure path formulas. Therefore, if $(T_1, s) \models \forall\psi$ and $T_2 \subseteq T_1$, it is

$$
\begin{array}{ll}
 & \forall \pi \in Path(T_1, s) \cdot (T_1, \pi) \models \psi \\
 & \qquad \left\{ \textsf{Lemma 1} \right\} \\
\Rightarrow & \forall \pi \in Path(T_2, s) \cdot (T_1, \pi) \models \psi \\
 & \qquad \left\{ \models \psi \textsf{ is the same for all paths} \right\} \\
\Rightarrow & \forall \pi \in Path(T_2, s) \cdot (T_2, \pi) \models \psi \\
 & \qquad \left\{ \textsf{Definition} \right\} \\
= & (T_2, s) \models \forall\psi
\end{array}
$$

and similarly for $\exists\psi$ and $T_1 \subseteq T_2$. QED.

Inductive application of this theorem is actually enough to produce the condition for weak preservation of the whole $\forall$CTL* and $\exists$CTL* logics.

Theorem 2 *Let T_1 and T_2 be transition systems and $T_1 \subseteq T_2$. Model checking T_2 weakly preserves $\forall CTL^*$ properties of T_1 , and model checking T_1 weakly preserves $\exists CTL^*$ properties of T_2 .*

Proof: We will show this by induction on $\forall$CTL* formulas. The induction will also use the path formula invariant that given a path $\pi \in Path(T_1, s)$ for an arbitrary state s and a $\forall$CTL* path formula ψ :

$$(T_2, \pi) \models \psi \Rightarrow (T_1, \pi) \models \psi$$

So let s be an arbitrary state and $\pi \in Path(T_1, s)$ be an arbitrary path of T_1 from that state.

Base case: If ϕ is a propositional formula, then

$$(T_1, s) \models \phi = (T_2, s) \models \phi = s \in \phi$$

therefore the case holds.

Inductive step for path formulas: Assume that for an arbitrary $\forall$CTL* formula ϕ' and for all states s it is:

$(T_2, s) \models \phi' \Rightarrow (T_1, s) \models \phi'$

Then:

$$\begin{array}{cl} & (T_2, \pi) \models \phi' \\ = & (T_2, s) \models \phi' \\ \Rightarrow & (T_1, s) \models \phi' \\ = & (T_1, \pi) \models \phi' \end{array}$$

and

$$\begin{array}{cl} & (T_2, \pi) \models \mathsf{X}\phi' \\ = & (T_2, \pi(1)) \models \phi' \\ \Rightarrow & (T_1, \pi(1)) \models \phi' \\ = & (T_1, \pi) \models \mathsf{X}\phi' \end{array}$$

Assume that for another arbitrary $\forall$CTL* formula ϕ'' the theorem holds. Then:

$$\begin{array}{cl} & (T_2, \pi) \models \phi' \mathsf{U} \phi'' \\ = & \exists i \in \mathbb{N} \cdot ((T_2, \pi(i)) \models \phi'' \wedge \forall j \in \mathbb{N}, j < i \cdot ((T_2, \pi(j)) \models \phi')) \\ & \qquad \left\{ \mathsf{Monotonicity} \right\} \\ \Rightarrow & \exists i \in \mathbb{N} \cdot ((T_1, \pi(i)) \models \phi'' \wedge \forall j \in \mathbb{N}, j < i \cdot ((T_1, \pi(j)) \models \phi')) \\ = & (T_1, \pi) \models \phi' \mathsf{U} \phi'' \end{array}$$

Lastly, assume that for two path $\forall$CTL* formulas, ψ' and ψ'' the theorem holds. Then for b being either & or | , and b' being respectively $\wedge$ and $\vee$:

$$\begin{array}{cl} & (T_2, \pi) \models b(\psi', \psi'') \\ = & b'((T_2, \pi) \models \psi', (T_2, \pi) \models \psi'') \\ & \qquad \left\{ \mathsf{Monotonicity} \right\} \\ \Rightarrow & b'((T_1, \pi) \models \psi', (T_1, \pi) \models \psi'') \\ = & (T_1, \pi) \models b(\psi', \psi'') \end{array}$$

Inductive step for states: Keeping the previous assumptions and definitions for convenience, we have:

$$\begin{array}{cl} & (T_2, s) \models b(\phi', \phi'') \\ = & b'((T_2, s) \models \phi', (T_2, s) \models \phi'') \\ & \qquad \left\{ \mathsf{Monotonicity} \right\} \\ \Rightarrow & b'((T_1, s) \models \phi', (T_1, s) \models \phi'') \\ = & (T_1, s) \models b(\phi', \phi'') \end{array}$$

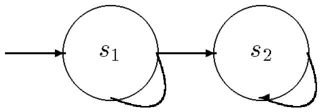
and

$$
\begin{array}{ll}
 & (T_2, s) \models \forall\psi' \\
= & \forall\pi \in Path(T_2, s) \cdot (T_2, \pi) \models \psi' \\
 & \qquad \{\text{Lemma 1}\} \\
\Rightarrow & \forall\pi \in Path(T_1, s) \cdot (T_2, \pi) \models \psi' \\
 & \qquad \{\text{Induction hypothesis, monotonicity}\} \\
\Rightarrow & \forall\pi \in Path(T_1, s) \cdot (T_1, \pi) \models \psi' \\
= & (T_1, s) \models \forall\psi'
\end{array}
$$

which concludes the inductive proof. The dual proof goes for the $\exists$CTL* case. QED.

4.2 Using two transition systems

Although both $\forall$CTL* and $\exists$CTL* are weakly preserved, we cannot use one transition system to preserve the full CTL*. For example let's take a look at the formula $\forall \mathsf{G} \exists \mathsf{F} p$. We are tempted to reason thus: since the formula talks about "all the paths" of a model, then if we consider two transition systems such that $T_1 \subseteq T_2$, then model checking of T_2 has to weakly preserve the property in T_1 , since all the paths in T_2 are already in T_1 . This is not true however; the paths *are there*, but may lose the existential property, as the trivial example that follows shows. Let $p = \{s_2\}$. Here the property $\forall \mathsf{G} \exists \mathsf{F} p$ holds for all paths s_1^ω (and also for $s_1^* s_2^\omega$) . But if we get rid of the transition s_1 to s_2 , then we still have the s_1^ω path, but it doesn't satisfy the property anymore.



To remedy this, consider the following: Given a transition system T , we can weakly preserve CTL* if we have *two transition systems* T_1 and T_2 such that

$T_1 \subseteq T \subseteq T_2$

in a way that, when we model check an existential subformula we use T_1 and when we model check a universal subformula we use T_2 . More formally, we define a new $\models$ relation (see [6]) for a model that has one initial state and two transition systems:

Definition 16 *For a model with two-transition systems* (T_1, T_2, s) *, the* $\models$ *relation is defined as follows:*

- *For path formulas it remains the same as for one transition system, e.g.*

$(T_1, T_2, \pi) \models \mathsf{X}\phi \; = \; (T_1, T_2, \pi(1)) \models \phi$

- *For propositional formulas it also remains the same*
- *For all disjunctions/conjunctions of state formulas it's the same*
- *For universal state formulas T_2-paths are used:*

$$(T_1, T_2, s) \models \forall\psi \;=\; \forall\pi \in Path(T_2, s) \cdot ((T_1, T_2, \pi) \models \psi)$$

- *For existential state formulas T_1-paths are used:*

$$(T_1, T_2, s) \models \exists\psi \;=\; \exists\pi \in Path(T_1, s) \cdot ((T_1, T_2, \pi) \models \psi)$$

Theorem 3 *If $T_1 \subseteq T \subseteq T_2$, then (T_1, T_2) weakly preserves CTL* for T , for all states.*

Proof: Let ϕ be a full CTL* formula. We will use structural induction on ϕ . If ϕ is a propositional formula or a disjunction or conjunction of formulas for which the theorem is true, then the theorem obviously holds for ϕ .

Now let the theorem hold for ϕ and ϕ' . Then for all $\pi \in Path(T, s)$:

$$\begin{array}{ll} & (T_1, T_2, \pi) \models \phi \\ = & (T_1, T_2, \pi(0)) \models \phi \\ \Rightarrow & (T, \pi(0)) \models \phi \\ = & (T, \pi) \models \phi \end{array}$$

and similarly for $\mathsf{X}\phi$ and $\phi \mathsf{U} \phi'$. Similarly also if ψ and ψ' are path formulas such that

$$(T_1, T_2, \pi) \models \psi \Rightarrow (T, \pi) \models \psi$$

and same for ψ' , then so are $\psi \& \psi'$ and $\psi \mid \psi'$.

Finally, assume that (for all paths π of T):

$$(T_1, T_2, \pi) \models \psi \Rightarrow (T, \pi) \models \psi$$

Then

$$\begin{array}{ll} & (T_1, T_2, s) \models \forall\psi \\ = & \forall\pi \in Path(T_2, s) \cdot ((T_1, T_2, \pi) \models \psi) \\ \Rightarrow & \forall\pi \in Path(T, s) \cdot ((T_1, T_2, \pi) \models \psi) \\ \Rightarrow & \forall\pi \in Path(T, s) \cdot ((T, \pi) \models \psi) \\ = & T \models \forall\psi \end{array}$$

and

$$
\begin{array}{ll}
 & (T_1, T_2, s) \models \exists\psi \\
= & \exists\pi \in Path(T_1, s) \cdot ((T_1, T_2, \pi) \models \psi) \\
\Rightarrow & \exists\pi \in Path(T, s) \cdot ((T_1, T_2, \pi) \models \psi) \\
\Rightarrow & \exists\pi \in Path(T, s) \cdot ((T, \pi) \models \psi) \\
= & T \models \exists\psi
\end{array}
$$

QED.

5 Model checking of abstract models

Now that we've met the two sufficient conditions for weak preservation in concrete models, we are going to explore what abstraction can do for us. We will first see an older framework [2] that managed to weakly preserve ∀CTL* properties and then, using our insight from the previous section we will see how Dams et. al [6] extend the work to the full CTL*. Both papers concern abstraction that is based on an abstract state partition.

5.1 Weak preservation of ∀CTL*

In [2], one abstract model $M_A = (I_A, T_A)$ whose states are those of an abstract state space $\mathcal{A}$ is built, for each concrete model $M = (I, T)$. For any two abstract states a and b

$$b \in T_A(a) = \exists s \in a, t \in b \cdot (t \in T(s))$$

and the initial state I_A is the only abstract state such that $I \in I_A$.

Definition 17 (Free transition system) *We will name the abstract transition system created above, the* free transition system *for T and $\mathcal{A}$.*

For this abstract model we will also define the "complete represented transition system". We will see that model checking a property for M_A is essentially equivalent to model checking with the complete transition system:

Definition 18 (Complete represented transition system) *Given the abstract transition system T_A , its complete represented transition system is a transition system T_C on concrete states that contains all possible transitions of T_A , i.e.*

$$\forall a \in \mathcal{A}, s \in a, b \in \mathcal{A}, t \in b \cdot (b \in T_A(a) = t \in T_C(s))$$

Obviously, if T is the original system from which we got T_A by abstraction, it is $T \subseteq T_C$.

Suppose now that we want to model check a formula ϕ compatible[3] with $\mathcal{A}$. As discussed above[4], we can get its abstract counterpart ϕ_A , if we substitute

[3] If our formula is not compatible, then this is not a good partition

[4] But going from the abstract to the concrete formula

all of its propositional subformulas with their abstract counterparts. For one propositional formula p the abstract counterpart will be:

$$p_A = \{a \in \mathcal{A} \mid a \subseteq p\}$$

The following theorem contains the essence of [2]:

Theorem 4 *If ϕ_A is a pure universal CTL* formula, then*

$$M_A \models \phi_A \;=\; (T_C, I) \models \phi$$

Proof: Let $s \in Reach(T_C, I)$ and $s \in a \in \mathcal{A}$. Let $\pi \in Path(T_C, I)$ and π_A be a path on abstract states such that for all natural i :

$$\pi(i) \in \pi_A(i)$$

There is such a path and it is unique, because $\mathcal{A}$ is a partition. The fact that π_A is a path of M_A is proved as follows: For any $i \in \mathbb{N}$:

$$\begin{array}{ll} & \quad \left\{\pi \in Path(T_C, I)\right\} \\ & \pi(i+1) \in T_C(\pi(i)) \\ & \quad \left\{\textsf{Definition of } T_C\right\} \\ = & \pi_A(i+1) \in T_A(\pi_A(i)) \end{array}$$

Let p_A be a propositional subformula of ϕ_A and p its concrete counterpart and a subformula of ϕ . Then

$$\begin{array}{ll} & (T_A, a) \models p_A \\ = & a \in p_A \\ & \quad \left\{\mathcal{A} \textsf{ is a partition}\right\} \\ = & a \subseteq p = \bigcup_{a' \in p_A} a' \\ & \quad \left\{\textsf{Set theory}\right\} \\ = & \forall s \in a \cdot (s \in p) \\ = & \forall s \in a \cdot ((T, s) \models p) \end{array}$$

A pure path formula will ultimately be either a propositional formula, or an application of X operator to a propositional formula, or an application of U operator to two propositional formulas, or a conjunction or disjunction of pure path formulas. All these operators are monotonic, that is provably, for any such path formula ψ_A :

$$\begin{array}{ll} & (T_A, \pi_A) \models \psi_A \\ = & \forall \pi \in \Pi_{\pi_A} \cdot ((T_C, \pi) \models \psi) \end{array}$$

where

$\Pi_{\pi_A} = \{\pi \in Path(T_C, T) \mid \forall i \in \mathbb{N} \cdot (\pi(i) \in \pi_A(i))\}$

We've shown before that for all $\pi \in Path(T_C, I)$ there is a path $\pi_A \in Path(T_A, I_A)$ such that $\pi \in \Pi_{\pi_A}$ and therefore:

$Path(T_C, I) = \bigcup_{\pi_A \in Path(T_A, I_A)} \Pi_{\pi_A}$

Thus we have for our pure universal formula $\phi_A = \forall \psi_A$:

$$
\begin{array}{ll}
 & (T_A, I_A) \models \forall \psi_A \\
= & \forall \pi_A \in Path(T_A, I_A) \cdot ((T_A, \pi_A) \models \psi_A) \\
= & \forall \pi_A \in Path(T_A, I_A), \pi \in \Pi_{\pi_A} \cdot ((T_C, \pi) \models \psi) \\
 & \qquad \left\{ Path(T_C, I) = \bigcup_{\pi_A \in Path(T_A, I_A)} \Pi_{\pi_A} \right\} \\
= & \forall \pi \in Path(T_C, I) \cdot ((T_C, \pi) \models \psi) \\
= & (T_C, I) \models \forall \psi
\end{array}
$$

QED.

The previous theorem together with theorem 2 has the major result of [2] as an immediate corrolary:

Theorem 5 *(Clarke, Grumberg, Long) If ϕ_A is a $\forall CTL^*$ formula then*

$M_A \models \phi_A \Rightarrow M \models \phi$

that is a weak preservation scheme for $\forall CTL^*$.

5.2 Weak preservation of CTL*

We show that abstraction helped us model check an "approximating" model in the concrete space, that was in the lattice of models "higher"[5] than the original model, thus achieving ∀CTL* weak preservation. By theorem 3, if we find an abstraction scheme that corresponds to model checking a "lower" model, we immediately have a scheme that weakly preserves the whole CTL*. This is exactly the solution presented in [6]. To demonstrate this solution, we introduce the notion of a simple represented model:

Definition 19 (Simple represented model) *Let $M_A = (I_A, T_A)$ be a model on the abstract state space $\mathcal{A}$. The concrete model $M = (I, T)$ is a* simple represented model *if and only if for any two states $a, b \in \mathcal{A}$, it is:*

$b \notin T_A(a) \Rightarrow \forall t \in b, s \in a \cdot (t \notin T(s))$

and if $b \in T_A(a)$ then for all $s \in a$ there is exactly *one state $t \in b$ such that $t \in T(s)$ and $I \in I_A$, formally:*

[5] In relation $T_1 \subseteq T_2$, T_2 is "higher" than T_1 .

$b \in T_A(a) \Rightarrow \exists t \in b \cdot \forall t' \in b \cdot (t = t' = t' \in T(s))$

Obviously, there always is at least one simple represented model.

An example of simple represented models is the following. Suppose that we have two abstract states $a = \{s_1, s_2\}$ and $b = \{s_3, s_4\}$ and $T_A = \{(a, b)\}$. The simple represented models of (a, T_A) are those whose initial state is either s_1 or s_2 and their transition system is one of the following

$$T_1 = \{(s_1, s_3), (s_2, s_3)\}$$

$$T_2 = \{(s_1, s_3), (s_2, s_4)\}$$

$$T_3 = \{(s_1, s_4), (s_2, s_3)\}$$

$$T_4 = \{(s_1, s_4), (s_2, s_4)\}$$

This definition is important because of the following theorem:

Theorem 6 *If ϕ_A is an existential pure CTL* formula, ϕ is its concrete counterpart and $M = (I, T)$ is a simple represented model of $M_A = (I_A, T_A)$, then*

$M_A \models \phi_A = M \models \phi$

Proof: By definition of a simple represented model, there is a one-to-one correspondence of paths of M and M_A . From $\pi \in Path(M)$ we can derive exactly one $\pi_A \in Path(M_A)$ such that

$\forall i \in \mathbb{N} \cdot (\pi(i) \in \pi_A(i))$

Since $\mathcal{A}$ is a partition, π_A is well-defined; we need only show that it is a path in M_A . But by the definition of a simple represented[6] model we have:

$\pi(i+1) \in T(\pi(i)) \Rightarrow \pi_A(i+1) \in T_A(\pi_A(i))$

therefore it is a path in M_A .

From $\pi_A \in Path(M_A)$ we can derive one path π such that

$\forall i \in \mathbb{N} \cdot (\pi(i) \in \pi_A(i))$

[6]This is the contrapositive of

$b \notin T_A(a) \Rightarrow \forall t \in b, s \in a \cdot (t \notin T(s))$

This path is built inductivelly: $\pi(0)$ is I . $\pi(1)$ is the unique state $t \in \pi_A(1)$ such that $t \in T(I)$. Likewise, at step $n+1$, $\pi(n+1)$ is the only state $t \in \pi_A(n+1)$ such that $t \in T(\pi(n))$.

We have shown that there is a one-to-one correspondence of paths. Furthermore, as shown in the proof of theorem 4, for a pure path property ψ_A and its concrete counterpart ψ_A and if the paths π and π_A correspond to each other according to the previous discussion, it is:

$$(T_A, \pi_A) \models \psi_A = (T, \pi) \models \psi$$

(Note that the property doesn't depend on the transition system at all).

Quantifying existentially over all paths doesn't change the situation. Let $\alpha(\pi)$ denote the abstract path corresponding to concrete path π :

$$\begin{array}{ll} & M_A \models \exists \psi_A \\ = & \exists \pi_A \in Path(M_A) \cdot ((T_A, \pi_A) \models \psi_A) \\ = & \exists \pi \in Path(M) \cdot ((T_A, \alpha(\pi)) \models \psi_A) \\ = & \exists \pi \in Path(M) \cdot ((T, \pi) \models \psi) \\ = & M \models \exists \psi \end{array}$$

QED.

As a consequence of this theorem, to weakly preserve the whole CTL* while model checking a concrete system $M = (I, T)$, we need an abstract transion T_A^C , such that for one of its simple represented models $M_S = (I, T_S)$, it is $T_S \subseteq T$. Such a transition system is given by Dams et al. [6]:

Definition 20 (Constrained transition system) *If T is a concrete transition system, the* constrained transition system *T_A^C for T and for the abstract state space $\mathcal{A}$ is such that*

$$b \in T_A^C(a) = \forall s \in a \cdot \exists t \in b \cdot (t \in T(s))$$

All we need to show now is

Theorem 7 *For model $M_A = (I_A, T_A^C)$ where $I \in I_A$ there is a simple represented model $M_S = (I, T_S)$, such that $T_S \subseteq T$.*

Proof: We will build T_S . Let $a \in \mathcal{A}$ and $b \in \mathcal{A}$. If $b \notin T_A^C(a)$, then there is no transition between concrete states in T_S . If $b \in T_A^C(a)$ it is

$$\forall s \in a \cdot \exists t \in b \cdot b \in T(a)$$

so for all s in a , we pick one state t whose existence is asserted by the above expression and define

$$t \in T_S(s)$$

and for all other $t' \in b$ we define

$t' \notin T_S(s)$

By construction T_S is a simple represented model of T_A^C and its transitions are a subset of T's transitions, therefore $T_S \subseteq T$. QED.

The end result of all the above, is the major result of [6] (also part of Dams's PhD thesis [5]):

Theorem 8 (Dams,Gerth,Grumberg) *Let T be a concrete transition system. Let T_A^F be the free transition system for T and $\mathcal{A}$ and let T_A^C be the concrete transition system for T and $\mathcal{A}$. Let $I \in \mathcal{S} \wedge I \in I_A \in \mathcal{A}$ and let ϕ_A be an abstract CTL* formula and ϕ its concrete counterpart. Then*

$(T_A^C, T_A^F, I_A) \models \phi_A \Rightarrow (T, I) \models \phi$

6 Concluding remarks

In this paper we have demonstrated an alternative way to study the problem of weak preservation and abstraction in model checking. In this context, we have reviewed two important papers in the area, [2] and [6]. We believe that this study conveys the basic understanding behind Denis Dams's work [6] and [5]. Of course in [6] and especially in [5] there is more interesting material that is not covered here. The most important aspect not covered here has to do with the practicality of the abstraction scheme, since a way is given to approximate the optimal free and constrained transition systems, which are in general hard to compute. One however can understand the feasibility of such a pursuit by the present paper: e.g. the free transition system may be approximated by any other transition system that is "higher" in the lattice of transition systems without compromising weak preservation.

The topic of weak preservation with abstract state space is more or less exhausted by the work of Dams and others. Other topics remain open to new research. It seems e.g. that preservation of weak CTL* would be more useful if combined with other methods of state reduction such as the composition of models. An interesting example of such research is [1]. And, to the author's opinion, no serious effort has been done yet to exploit results of formal reasoning in other paradigms of programming theory, such as refinement.

Our study doesn't strive to achieve any originality. Results about weak preservation in concrete space are of no practical interest whatsoever, since there are trivial solutions. If we are to use two transition systems, a trivial and completely useless solution would be to use the top and bottom of the lattice. The problem of weak preservation is to find better approximations, that is, a transition system that is higher than the original but as low as possible and a transition system that is lower than the original but as high as possible. And of course if we use the original transition system we trivially get the best result.

Interest from a practical point of view only kicks in when we introduce abstraction schemes. An abstraction scheme imposes constraints on which systems in the concrete lattice we can model check, thus finding the best approximation that the abstraction permits becomes an interesting problem.

Theoretically however, the study of CTL* reveals an important property: CTL* is built by monotonic primitive formulas but by itself it is not monotonic. This lack of monotonicity stems from the fact that the monotonic primitives are dual (we have both universal and existential path quantification) and thus incompatible. This is interesting with respect to programming theory, especially refinement-oriented (see e.g. [7]) that typically assumes that its primitive constructs share the same monotonicity[7]. Now note that we can always take advantage of the monotonicity of parts of CTL* in our proofs, like we did in this paper. Since this works out in model checking, it can serve as an inspiration if we are to undertake the extension of refinement calculi to fit existential path quantifications[8].

References

[1] R. Alur, L. deAlfaro, T. A. Henzinger, and F. Y. C. Mang. Automating modular verification. In *CONCUR '99: Concurrency Theory, Tenth Int. Conference*, volume 1664 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, 1999.

[2] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Prog. Lang. and Systems*, 16(5):1512–1542, 1994.

[3] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[4] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, 1977.

[5] D. Dams. *Abstract interpretation and partition refinement for model checking*. PhD thesis, Technische Universiteit Eindhoven, 1996.

[6] D. R. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Prog. Lang. and Systems*, 19(2):253–291, 1997.

[7] E. C. R. Hehner. *A Practical Theory of Programming*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.

[7] Refinement calculi have a preference for universal quantification and thus are compatible and comparable with $\forall$CTL* or its most commonly used sublogic LTL

[8] Technically, refinement calculi do not work with execution paths, but we *can* draw precise analogies between the temporal logic and the refinement paradigm