Revising Specifications with CTL Properties using Bounded Model Checking

No Author Given

No Institute Given

Abstract. During the process of software development, it is very common that inconsistencies arise between the formal specification and some desired property. Belief Revision deals with the problem of accommodating new information that may be inconsistent with an existing knowledge base.

In this paper, we propose the use of belief revision techniques in order to deal with inconsistencies in formal specifications. The main problem to be solved is that the most well known results for belief revision only hold for logics which are monotonic and compact, while most discrete-time temporal logics used to express system properties – and in particular, CTL — are not compact. We suggest the use of bounded model-checking, transforming the problem from CTL into classical propositional logic and then transforming back the results to suggest revisions to the user.

Keywords: Model-checking, belief revision, formal specification, CTL.

1 Introduction

A system specification evolves when there is a conflict between the actual properties of the system and its intended new behaviour. In terms of formal specifications, this can be modelled by the existence of an inconsistency between an existing specification and a desired property.

Handling inconsistencies in specifications is a critical activity in the software development process. A variety of techniques has been developed for checking specifications for inconsistencies. These include formal techniques such as those based on model checking or theorem proving [1–5]. Model checking of specifications, in particular, is usually performed in CTL logic or in some related formalism.

While many of these approaches provide rigorous, and often automated, analysis of software specifications to reveal inconsistencies, they often also do not support the system developer in solving these inconsistencies after they have been discovered.

To address this issue, a first proposal of evolving system specifications in CTL was presented in [6] based on techniques of belief revision to suggest ways for changing system specifications, as illustrated in Figure 1. Belief revision [7, 8] is a sub-area of artificial intelligence whose main focus is to keep the consistency of a set of beliefs when new beliefs are incorporated.

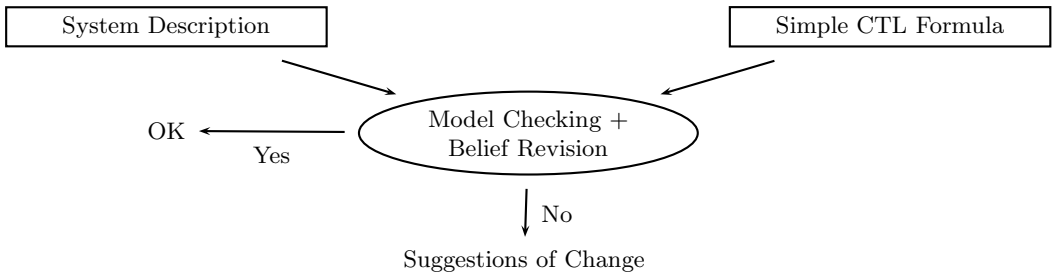

Fig. 1. Model-checking with belief revision

Belief revision is normally done in terms of *revision operations*. However, it is not always the case that such operations exist for all formalisms. The method presented in [6] could only deal with simple CTL expressions, but typically one would light to check complex system properties. In the case of revising a specification with an arbitrarily complex property, it is very important to determine if a specification can be revised at all.

In this work, we study CTL specification and model checking techniques with respect to obtaining a method that guarantees that revised specifications always exist. In particular, we study the existence of *contraction* operations over specifications, which is the belief revision operation that performs the removal of some currently held belief, that is, some current system property that follows from the specification. The addition of new properties can be done by first contracting its negation, and then simply adding the new property to the contracted specification.

The starting point of this study is a quite general result on the existence of contraction operations for formal systems [9]. It turns out that CTL (and any other temporal logic used for model checking such as LTL, CTL and CTL*) *does not fulfil the required conditions* to guarantee the existence of a contracted theory. This means that, in general, one cannot guarantee that a revised specification exists using the traditional tools of belief revision.

The main contribution of this work is an interaction between model checking and bounded model checking that helps us solve this problem. Our proposal is illustrated is Figure 2.

The idea is that, once an inconsistency has been detected in a specification by model checking, one can determine a *bound* for *bounded model checking* [10]. The technique of bounded model checking transforms the model checking problem into a classical satisfiability problem. As classical logic satisfies the conditions of existence of a revised specification [9], one then performs a contraction over the transformed specification, and finally one has to work the contraction back to obtain the suggestions on how to revise the original specification.

A second advantage of this approach is that, with the translation into classical logic, CTL statements with arbitrary complexity can be accepted for model

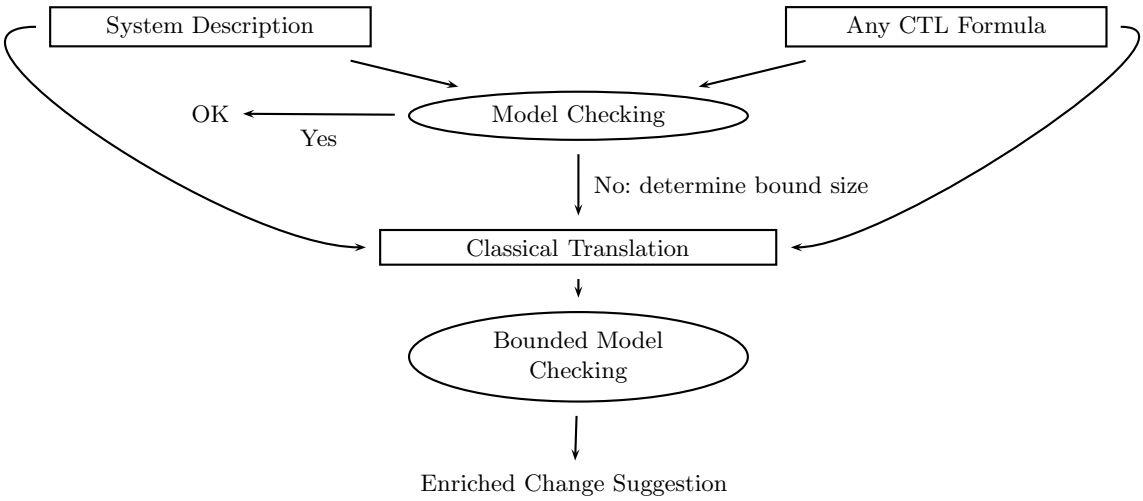

Fig. 2. Specification Revision via Bounded Model Checking

checking and specification revision, so a new kind of change suggestions can be obtained, enriching the capabilities of the method.

The paper is organised as follows: in the next two sections, we briefly introduce the areas of Belief Revision and CTL Model Checking. In Section 4 we discuss the applicability of standard belief revision to CTL. Next in Section 5 we present our method. Then we conclude, pointing towards future work.

2 Belief Revision

The necessity to model the behaviour of dynamic knowledge bases formed the basis of belief revision theory [7, 8]. Most of the literature in the area is based on the seminal work of Alchourrón, Gärdenfors and Makinson [11], that have proposed some postulates that describe the formal properties that a revision process should obey. They have also proposed some constructions that satisfy the postulates. The theory became known as the AGM paradigm, due to the initial of the authors.

In AGM theory, the beliefs of an agent are represented by a belief set, a set of formulas closed under logical consequence ($K = Cn(K)$, where Cn is a supraclassical consequence operator). There are three types of change operators for a belief set (K). In *expansion* ($K + \alpha$), a consistent information α, together with its logical consequences, is added to the belief set K. In *contraction* ($K - \alpha$), the information α is abandoned. Since the set K is logically closed, it could be necessary to abandon other beliefs that would imply α. In *revision* ($K * \alpha$), an information α is added to K and to keep consistency it can be necessary to abandon other beliefs of K.

Besides belief sets, one can use the idea of possible worlds in order to represent the beliefs of the agent. Possible worlds can be thought of as possible states of the world (or, in propositional logics, propositional valuations). Given a belief set K, $[K]$ is the set of the possible worlds where all formulas of K are true. And for a set W_k of possible worlds we can define a corresponding belief set K as the set of formulas that are true in all the worlds of W_k.

This was the line followed in [6]: the possible worlds were associated to the possible states of the system and an operation of revision was proposed based on [12].

In this work, we follow a different approach to belief revision. Instead of dealing with belief sets, i.e., sets closed under logical consequences, we are interested in finite specifications. We use the idea of belief bases, in the line of [8].

3 CTL Model-Checking

The basic principle of model-checking is: given a property of the system, described in a temporal logic, determine whether the finite state machine that represents the described system satisfies such property. In other words, verify whether a formula f is true in a graph G of states.

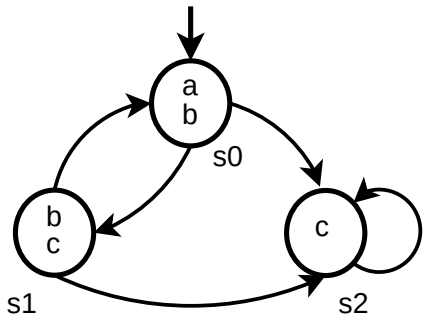

Fig. 3. Kripke Structure

The finite state machine that represents the system can be described as a specific Kripke structure, that is defined as: a set S of states, a set R of transitions between states (where each state must have a successor), a set I of initial states and a function L that associates to each state a set of propositions that hold in that state. To model a state in *deadlock* (state without successors) it suffices to create a transition from the state to itself. Figure 3 represents a Kripke structure where $P = \{a, b, c\}$, $S = \{s0, s1, s2\}$, $R = \{(s0, s1), (s0, s2), (s1, s0), (s1, s2), (s2, s2)\}$, $I = \{s0\}$ with $L(s0) = \{a, b\}$, $L(s1) = \{b, c\}$ and $L(s2) = \{c\}$.

3.1 Computation Tree Logic

The properties of the system are described in temporal logic. Temporal logic is a type of modal logic where it is possible to represent and to reason about propositions related to time. Through temporal logic it is possible to express sentences of the type "I am *ALWAYS* hungry" or "I will be hungry *UNTIL* I eat in a all-you-can-eat restaurant".

Clarke and Emerson [13] proposed CTL (Computation Tree Logic), a logic capable to consider different possible futures, through the notion of branching time. The idea of this logic is to quantify over the possible runs of a program through the notion of paths that exist in the space of states of the system. The properties can be evaluated with respect to all the runs or some run. This logic is used in some model-checkers, such as NuSMV, which we have used in our implementation. The syntax of CTL is given by the following definition:

$$\phi ::= p|\neg\phi|\phi \wedge \phi|(AX\phi)|(AG\phi)|(AF\phi)|(EX\phi)|(EG\phi)|(EF\phi)|E(\phi\ U\phi)|A(\phi\ U\phi)$$

where p is a propositional atom, $\neg$, $\wedge$ are the usual logical connectives and the other are temporal operators. Each temporal operator is composed of a path quantifier (E, "there exists a path", or A, "for all paths") followed by a state operator (X, next state in the path, U, until, G, globally, or F, finally). CTL has the following semantic definition:

Definition 1. *Let M be a Kripke structure and $\pi(i)$ the i-th state of a path. We say that $M, s \models \phi$ if and only if ϕ is true in the state s of M. Thus we have:*

1. $M, s \models p$ *iff* $p \in L(s_0)$
2. $M, s \models \neg\phi$ *iff* $M, s \not\models \phi$
3. $M, s \models \phi_1 \wedge \phi_2$ *iff* $M, s \models \phi_1$ *and* $M, s \models \phi_2$
4. $M, s \models EX\ \phi$ *iff there is a state* s' *of* M *such that* $(s, s') \in R$ *and* $M, s' \models \phi$
5. $M, s \models EG\ \phi$ *iff there is a path* π *of* M *such that* $\pi(1) = s$ *and* $\forall i \geq 1 \bullet M,\pi(i) \models \phi$
6. $M, s \models E(\phi_1\ U\ \phi_2)$ *iff there is a path* π *of* M *such that* $\pi(1) = s$ *and* $\exists i \geq 1 \bullet (M, \pi(i) \models \phi_2 \wedge \forall j, i > j \geq 1 \bullet M,\pi(j) \models \phi_1)$

The other temporal operators can be derived from EX, EG and EU:
AX $\phi = \neg$ EX $\neg\phi$
AG $\phi = \neg$ EF $\neg\phi$
AF $\phi = \neg$ EG $\neg\phi$
EF ϕ = E [true U ϕ]
A[ϕ U β] = $\neg$ E[$\neg\beta$ U $\neg\phi \wedge \neg\beta$] $\wedge \neg$ EG $\neg\beta$

We say that a Kripke structure M satisfies a formula CTL ϕ if $M, s_0 \models \phi$, where s_0 is an initial state.

The main approach for automatic formula verification in CTL is based on the fixed point theory to characterize its operators. For details the reader is referred to [14].

4 Guaranteeing the Existence of Specification Revisions

The AGM framework for belief revision was originally formulated having classical logic in mind. There have been some attempts to apply the operations for different logics, such as modal or description logics, but in [15] it was shown that AGM can only be applied to logics that are decomposable, i.e., logics for which it holds that for any sets of formulas X, K if $Cn(\emptyset) \subset Cn(X) \subset Cn(K)$, then there exists a set Y such that $Cn(Y) \subset Cn(K)$ and $Cn(X \cup Y) = Cn(K)$. Flouris has shown that some important description logics are not decomposable and therefore, do not admit an AGM style contraction operation.

CTL (and modal logics in general) are decomposable. This means that AGM-style operations can be applied. However, the use of logically closed sets leads to problems from the computational point of view, since they are typically infinite.

For belief bases, [9] has shown that if the logic is compact and monotonic, the typical constructions can be applied. But CTL and other temporal logics based on discrete time are not compact. This means that it is not possible to directly inherit all the results, as the theorems in [9] do not say anything about the case where the logic is not compact.

The approach we follow here is to avoid the issue by first translating the problem into propositional logic and then applying standard methods of belief revision.

5 Revising via Bounded Model Checking

A solution to the problem of ensuring the applicability of belief revision techniques is proposed here. This is achieved by means of Bounded Model Checking and its associated translation of both specification and CTL property to propositional logic. The process is illustrated in Figure 2 and detailed next.

5.1 Bounded Model Checking

The idea of Bounded Model Checking [10, 16] is to fix a bound k for the maximal path in a Kripke Model that can be traversed. This allows a given Kripke Model K to be translated into a propostional classical formula K^k.

The semantic of CTL formulas is then altered to guarantee that the size of paths traversed in the evaluation of a formula are never larger than k. A special model-theoretic construct is needed for the semantics of the EG operator, namely the k-loop, that is a path of size at most k containing a loop. The k-bounded semantic of CTL formulas, $M, s \models^k \phi$, $k \geq 0$, is defined as

1. $M, s \models^k p$ iff $p \in L(s_0)$
2. $M, s \models^k \neg\phi$ iff $M, s \not\models^k \phi$
3. $M, s \models^k \phi_1 \wedge \phi_2$ iff $M, s \models^k \phi_1$ and $M, s \models^k \phi_2$
4. $M, s \models^k \text{EX}\, \phi$ iff there is a state s' of M such that $(s, s') \in R$ and $M, s' \models^{k-1} \phi$

5. $M, s \models^k \text{EG } \phi$ iff there is a k-loop π of M such that $\pi(1) = s$ and $\forall i \geq 1 \bullet M,\pi(i) \models^{k-i} \phi$
6. $M, s \models^k \text{E}(\phi_1 \text{ U } \phi_2)$ iff there is a path π of M such that $\pi(1) = s$ and $\exists i \geq 1 \bullet (M, \pi(i) \models^{k-i} \phi_2 \wedge \forall j,i > j \geq 1 \bullet M,\pi(j) \models^{k-j} \phi_1)$

With a fixed bound k and this k-bounded semantics, one can translate a bounded CTL formula ϕ into a classical formula A^k_ϕ. One then submits the formula $K^k \wedge A^k_\phi$ to some efficient SAT solver (that is, a classical propositional theorem prover, such as Chaff[17] or Berkmin [18]) to obtain a verdict. If the formula is unsatisfiable, this means that $\neg\phi$ holds at the Kripke Model K. Otherwise a classical valuation v is presented by the SAT solver, which represents a path in traversed the model such that ϕ holds at the Kripke model.

5.2 The Method of BMC Revisions

By checking a formula ϕ against a model, one is trying to force the property $\neg\phi$ over the revised model. In fact, if K and ϕ are inconsistent, this means that K "implies" $\neg\phi$. So if $K^k \wedge A^k_\phi$ is inconsistent, nothing needs to be done, and this is our ideal case.

The method for obtaining change suggestions from bounded model checking is triggered in the situation where the SAT solver outputs a valuation that satisfies the formula $K^k \wedge A^k_\phi$, as represented in Figure 4.

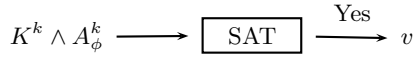

Fig. 4. Bounded Model Checking Triggering Belief Revision

Any classical valuation v can be directly transformed into a formula $A_v = \bigwedge_{v(ell_i)=1} \ell_i$ that conjoins all literals satisfied by v.

The basic idea of the method is to revise the specification represented by $K^k \wedge A^k_\phi$ so that valuation v is not allowed. However, it may be the case that $K^k \wedge A^k_\phi \wedge \neg A_v$ may be a satisfiable formula, satisfied by valuation v_2, so that $\neg A_{v_2}$ is added to the theory. This process is iterated m times until an inconsistent theory K' is obtained:

$$K' = K^k \wedge A^k_\phi \wedge \neg A_{v_1} \wedge \ldots \neg A_{v_m}$$

As the number of possible valuations is finite, the process always terminates. The revised specification K_{rev} is obtained by

$$K_{\text{rev}} = K' * \neg A^k_\phi = (K' - \neg A^k_\phi) + \neg A^k_\phi$$

One last step is still missing, namely, the translation back from propositional classical K_{rev} into change suggestions in the original specification K.

Theorem 1. *The revised specification K_{rev} always exists.*

This means that the initial specification can be revised with respect to any CTL formula *provided a large enough bound k is used*. One possible way of determining such bound is to run a usual (unbounded) model checking with respect to the desired property, $\neg\phi$.

This method is apparently trading a PSPACE-complete problem (model checking) for an NP-complete one (SAT). Is there a price to pay for it? In fact:

- No upper limit has been established over m, the number of times that SAT is executed; in the worst case, it is an exponential number, in terms of the number of propositional variables.
- The bound k may be hard to find. In particular, an approximate approach may be used via bounded model checking.

5.3 Generating Change Suggestions

To generate change suggestions, one has to compare the initial translation K^k with the revised theory K_{rev} and propagate back the differences to the specification. The translation back is not a problem because:

- The atomic formulas of K all represent clear facts about the specification such as "p holds at state S" and "S_0 accesses S_1".
- Belief revision techniques usually do not introduce new atomic symbols, so K_{rev} is a boolean combination of the same literals present in K.

As K_{rev} can be simply translated back in terms of the original specification elements, the only problem now is how to interpret that revised specification.

In general, the initial specification K is a *definite description*, presented as a conjunction of literals, such as "p holds at state S_0 which accesses state S_1, where p is false". In contrast, the revised model K_{rev} will very likely have parts of it constituted of the disjunction of several options, eg. statements of the form "state S_0 accesses either S_i or S_j and p either holds in both S_i and S_j, or in none of them".

Once the differences between K and K_{rev} have been detected, one may present them to the user as *the* output solution. However, what one needs in the end of the revision process is a definite model again, so one is faced with the following possibilities:

- Present the possible definite models in some (perhaps arbitrary) order, such that all presented definite models are the only possible ones entailed by the specification; there may be an exponentially large number of such models.
- Choose a single definite model to present as a suggestion. The preference of one model over the others may be built-in in the revision method, so that K_{rev} is in fact a definite description.

We believe that end-users must have a say on which kind of suggestion they find more useful.

6 Conclusions

We have shown how belief revision can be used to revise specifications even when the properties to introduce are specified in CTL. The technical problem that arises is that one cannot guarantee the existence of a revised model in CTL and similar discrete-time temporal logics, such as LTL and CTL*.

The solution comes in using Bounded Model Checking and a translation of both specification and CTL formulas to propositional logic, such that a SAT solver is applicable. A method was proposed to iterate through these theories until belief revision is applicable, and the revised version is guaranteed to exist, and easily translatable back to the specification level.

Future works include studying the formal properties of the proposed method and implementing such specification revision operations. In particular, a process of stepwise refinements of a specification is to be submitted to such a process. The feedback from such a process will help us decide on the best way to present to the specifier the change suggestions provided by the method.

References

1. Winter, K.: Model checking for abstract state machines. Journal of Universal Computer Science **3**(5) (1997) 689–701
2. Bessow, R.: Model Checking Combined Z and Statechart Specifications. PhD thesis, Technical University of Berlin, Faculty of Computer Science (November 2003) available from http://edocs.tu-berlin.de/diss/2003/buessow_robert.pdf.
3. Leuschel, M., Butler, M.: ProB: A model checker for B. In Araki, K., Gnesi, S., Mandrioli, D., eds.: FME 2003: Formal Methods. LNCS 2805. Springer-Verlag (2003) 855–874
4. Kolyang, Santen, T., Wolff, B.: A structure preserving encoding of Z in Isabelle/HOL. In von Wright, J., Grundy, J., Harrison, J., eds.: Proceedings of the 9th International Conference on Theorem Proving in Higher Order Logics, Turku, Finland, Springer-Verlag LNCS 1125 (1996) 283–298
5. E.M. Clarke, O. Grumberg, K. Hamaguchi: Another look at LTL model checking. In D. Dill, ed.: Proceedings of the 6th International Conference on Computer-Aided Verification CAV. LNCS 818. Springer-Verlag (1994) 415–427
6. de Sousa, T.C., Wassermann, R.: Handling inconsistencies in CTL model-checking using belief revision. In: Proceedings of the Brazilian Symposium on Formal Methods (SBMF). (2007)
7. Grdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. MIT Press (1988)
8. Hansson, S.O.: A Textbook of Belief Dynamics. Kluwer Academic Publishers (1997)
9. Hansson, S.O., Wassermann, R.: Local change. Studia Logica **70**(1) (2002) 49–76
10. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without bdds. In Cleaveland, R., ed.: Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99. Lecture Notes in Computer Science, Springer (1999) 193–207
11. Alchourron, C., Grdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic **50**(2) (1985) 510–530

12. Grove, A.: Two modellings for theory change. Journal of Philosophical Logic **17**(2) (1988) 157–170
13. Clark, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In Kozen, D., ed.: Logics of Programs. LNCS 131. Springer-Verlag (1981) 52–71
14. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press (2000)
15. Flouris, G.: On Belief Change and Ontology Evolution. PhD thesis, University of Crete (2006)
16. Penczek, W., Wozna, B., Zbrzezny, A.: Bounded model checking for the universal fragment of ctl. Fundamenta Informaticae **51**(1) (2002) 135–156
17. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: Proceedings of the 38th Design Automation Conference (DAC'01). (2001) 530–535
18. Goldberg, E., Novikov, Y.: Berkmin: A Fast and Robust SAT Solver. In: Design Automation and Test in Europe (DATE2002). (2002) 142–149