CTL May Be Ambiguous when Model Checking Moore Machines

Cédric Roux and Emmanuelle Encrenaz

UPMC – LIP6 – ASIM
12, rue Cuvier, 75252 Paris CEDEX 5 – France
{Cedric.Roux, Emmanuelle.Encrenaz}@lip6.fr

Abstract. The model checking problem is defined over Kripke structures. However, hardware designers often handle other models, such as Moore machines. When model checking their designs using CTL as a logic, they must translate them into Kripke structures. A given CTL property may be believed to be true (conversely false) over the Moore machine and in fact be false (conversely true) on the derived Kripke structure. This may lead to ambiguities if the designer does not fully understand the translation scheme he uses, which may be the case if he uses automatic tools. We present iCTL, a logic specifically designed to work with Moore machines, which extends CTL to help the designer removing possible ambiguities when model checking Moore machines. We show that it is strictly more expressive than CTL.

1 Introduction

While developing a symbolic model checker to verify hardware systems described as a composition of synchronous Moore machines, we came across an interesting problem. We use CTL [2] as logic and the formulae we want to verify may include values of input signals of the Moore machines. These input signals do label the transitions of the Moore machine. Since CTL is defined over Kripke structures and not Moore machines, and because the transitions of Kripke structures are not labelled, when translating a Moore machine into a Kripke structure, one has to integrate the input signals in the states of the Kripke structure. Several choices are possible. Depending on the translation chosen, the truth value of a given property may either be true or false over the derived Kripke structure. This introduces an ambiguity that the designer must be aware of when verifying his designs. He has to know how his model is translated into the one used by the model checker, and has to write properties with this in mind, so not to get confused by the answer of the tool. Not doing so could even lead to a counter-intuitive situation, where the designer might view his model as being buggy where in fact he simply wrote wrong formulae, thinking them over Moore machines and not over the derived Kripke structures.

In [3] the authors translate a Moore machine into a Kripke structure by incorporating the input configurations in the source state of the transitions. And they define the truth value of a CTL property over a Moore machine as being the truth value of this property over the Kripke structure. We think that such an approach leads to ambiguities.

In SMV [5], one directly writes Kripke structures and CTL formulae over these structures. It is possible to create free variables (that may represent input signals of Moore machines incorporated into the current state of the Kripke structure). This leads to exactly the same situation as [3].

The VIS model checker [1] accepts, among others, systems described in a Verilog subset, in which collections of Moore machines can be represented. It supports modularity and the concept of input and output signals is present. However, an input signal can appear in a CTL formula only if it is declared of type `reg`, which forces its assignment in guarded blocks. As a consequence, depending on the way this assignment is done, input signals of a Moore machine will be included into the source or target state of the transitions in the Kripke structure, which influences the results of the verification of a given formula.

The purpose of this article is to suggest to add two new operators to CTL to bring together the intuitive idea one can have regarding the truth value of a formula over the Moore machine and the one obtained by the verification algorithm over the Kripke structure. These two operators are specifically designed to handle Kripke structures derived from Moore machines and are meaningless in other cases. We hope that their utilization will facilitate the writing of formulae and the understanding of the results produced by the model checker.

2 Translating a Moore Machine into a Kripke Structure

Several translation schemes from a Moore machine into a Kripke structure are possible. The simplest one is to remove the inputs labelling transitions. Since we want to express properties including input signals, we abandoned such a scheme. Another way is to put the input signals into the target state of a transition. Since we plan to compose Moore machines, this solution can't be retained because the outputs of one machine which are inputs of one other have to have the same temporal behavior as the other inputs of the second machine. So, we have to put the inputs into the source state of a transition.

Here follows the formal definitions of Kripke structures, Moore machines, and the translation scheme we adopted.

Definition 1. *A Kripke structure is a five-tuple* $\langle S, S_0, P, \mathcal{L}, R \rangle$ *where*

1. S *is a finite set of states,*
2. $S_0 \subseteq S$ *is the set of initial states,*
3. P *is a finite set of atomic propositions (we define* $n_P = |P|$*),*
4. $\mathcal{L} = \{l_0, \ldots, l_{n_P - 1}\}$ *is a vector of* n_P *functions, each function defining the value of exactly one atomic proposition; for all* $0 \leq i \leq n_P - 1$ *we have* $l_i : S \to \mathbb{B}$*; for all* $s \in S$*, we have that* $l_i(s)$ *is true iff the atomic proposition associated to* l_i *is true in* s*,*
5. $R \subseteq S \times S$ *is the transition relation.*

Definition 2. *A Moore machine is a structure* $\langle S, S_0, I, O, \mathcal{L}, R \rangle$ *where*

1. S *is a finite set of states,*
2. $S_0 \subseteq S$ *is the set of initial states,*
3. I *is the finite set of input symbols,*
4. O *is the finite set of output symbols (we define* $n_O = |O|$*),*
5. $\mathcal{L} = \{l_0, \ldots, l_{n_O - 1}\}$ *is a vector of* n_O *functions, each function defining the value of exactly one output symbol; for all* $0 \leq i \leq n_O - 1$ *we have* $l_i : S \rightarrow \mathbb{B}$*; for all* $s \in S$*, we have that* $l_i(s)$ *will be true iff the output symbol associated to* l_i *is true in the state* s*,*
6. $R \subseteq S \times 2^I \times S$ *is the transition relation.*

The Moore machines we handle are complete and deterministic. Complete means that each state has one successor for any input configuration. Deterministic means that for a given input configuration, a state s will always lead to the same state s'.

Definition 3. Translating a Moore Machine by Putting the Inputs in the Source State *Given a Moore machine* $\langle S_M, S_{M0}, I_M, O_M, \mathcal{L}_M, R_M \rangle$*, we deduce the Kripke structure* $\langle S_K, S_{K0}, P_K, \mathcal{L}_K, R_K \rangle$ *where:*

- $S_K = S_M \times 2^{I_M}$,
- $S_{K0} = S_{M0} \times 2^{I_M}$,
- $P_K = I_M \cup O_M$ *(we define* $n_{I_M} = |I_M|$ *and* $n_{O_M} = |O_M|$*),*
- $\mathcal{L}_K = \{l_{O0}, \ldots, l_{On_{O_M}-1}\} \cdot \{l_{I0}, \ldots, l_{In_{I_M}-1}\}$*; for all* $0 \leq i \leq n_{O_M} - 1$*, we have* $l_{Oi} : S_K \rightarrow \mathbb{B}$*; for all* i*, for all* $s = (s_1, c_1) \in S_K$*, we have that* $l_{Oi}(s)$ *is true iff* $l_{Mi}(s_1)$ *is true; for all* $0 \leq i \leq n_{I_M} - 1$*, we have* $l_{Ii} : S_K \rightarrow \mathbb{B}$ *(each* l_{Ii} *is associated to one and only one input signal); for all* i*, for all* $s = (s_1, c_1) \in S_K$*, we have that* $l_{Ii}(s)$ *is true iff the component of* c_1 *corresponding to the input signal associated to* l_{Ii} *is true,*
- $R_K \subseteq S_K \times S_K$ *and* $\forall\, (s, c_i) \in S_K, \forall\, (s', c'_i) \in S_K$*, we have* $((s, c_i), (s', c'_i)) \in R_K$ *iff* $(s, c_i, s') \in R_M$.

An example of a trivial Moore machine and the derived Kripke structure is shown in figure 1.

3 A disturbing example

We could simply state that a CTL formula is true in a Moore machine if and only if it is true in the corresponding Kripke structure as done in [3] but the verification results obtained may disturb the designer.

As an illustration, we propose to check the CTL property ($\mathbf{EX}\ p$) $\wedge$ ($\mathbf{EX}\ \neg p$) over the Moore machine depicted on figure 1.

This formula would be true on a Kripke structure obtained from the Moore machine by removing the inputs, but it is false on the Kripke structure shown on figure 1 (which is the one obtained with the translation of definition 3), because neither A_0 nor A_1 has a successor verifying $\neg p$ and a successor verifying p.

In fact, the formula ($\mathbf{EX}\ p$) $\wedge$ ($\mathbf{EX}\ \neg p$) is ambiguous over the Moore machine: do we mean that both successors are selected by the same input configuration or by different input configurations?

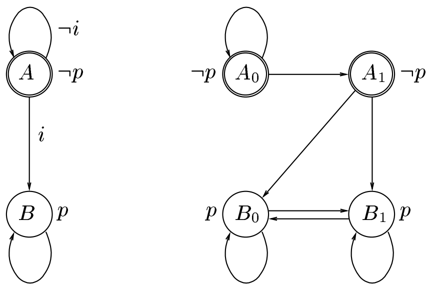

Fig. 1: A trivial Moore machine and its derived Kripke structure

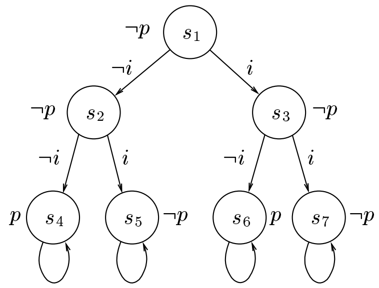

Fig. 2: A Moore machine illustrating the use of $\exists_I$ and $\forall_I$

4 *i*CTL – CTL Model Checking with Input Configurations

We introduce two new operators to CTL. These two operators are $\forall_I$ and $\exists_I$. This defines a new logic, that we call iCTL. Given ϕ, an iCTL formula (that may contain $\forall_I$ and $\exists_I$ operators), $\forall_I \phi$ stands for "for all input configuration, ϕ holds" and $\exists_I \phi$ stands for "there is an input configuration for which ϕ holds".

Here follows the formal definition of iCTL.

4.1 Syntax and Semantics of *i*CTL

The syntax is the same as the one of CTL, with the following added rule for state formulae.

– if f is a state formula, then $\forall_I f$ and $\exists_I f$ are state formulae.

The semantics remains the same, with the following added rules.

As the two new operators deal with input configurations, the Kripke structure they apply on are the ones given by our translation from Moore machines. The symbols are thus the same than those from definition 3.

$$M, s \models \forall_I f \Leftrightarrow s = (s_M, c_M) \text{ and for all } c'_M \in 2^{I_M},\ s' = (s_M, c'_M) \text{ and we have that } s' \models f,$$
$$M, s \models \exists_I f \Leftrightarrow s = (s_M, c_M) \text{ and for one } c'_M \in 2^{I_M},\ s' = (s_M, c'_M) \text{ and we have that } s' \models f.$$

Since our Moore machines are complete, for all input configurations, the state s' exists in the Kripke structure, thus $s' \models f$ is sound.

Using iCTL, we now can define when a Moore machine validates a logical formula.

Definition 4. *A Moore machine M validates a formula f of iCTL if and only if the formula is true in the corresponding Kripke structure, as given by the transformation of definition 3.*

This definition is the same as in [3], but we expect the designer to remove the ambiguities of CTL by using $\exists_I$ and $\forall_I$ in the places where they are needed.

4.2 Examples

The Moore machine of figure 2 will be used as example.

On the Kripke structure derived from it by the translation of definition 3, we've got that the formula **AX EX** p is false in $s_1.\bar{i}$ and $s_1.i$. Looking at the Moore machine, one might think that this formula is true in s_1, since all its successors have a successor where p is true (states s_4 and s_6). The formula **AX** ($\exists_I$ (**EX** p)) is true in $s_1.\bar{i}$ and $s_1.i$ on the derived Kripke structure. This corresponds to the intuition one might have about the truth value of **AX EX** p over the Moore machine. We see here that to capture this intuition, $\exists_I$ is necessary.

Similarly, the formula **EX AX** p is true in $s_1.\bar{i}$ and $s_1.i$ in the derived Kripke structure while **EX** ($\forall_I$(**AX** p)) is false in $s_1.\bar{i}$ and $s_1.i$. This latest interpretation seems to be consistent with the intuition that one might have for the truth value of **EX AX** p in the state s_1 of the Moore machine.

4.3 iCTL Is More Expressive than CTL

Given a formula $f \in i$CTL and a formula $g \in$ CTL, we say that f is equivalent to g if and only if for all Kripke structure K derived from a Moore machine M using the translation of definition 3, for all state s of K, we have that $K, s \models f$ iff $K, s \models g$. (This is the global equivalence of [4].)

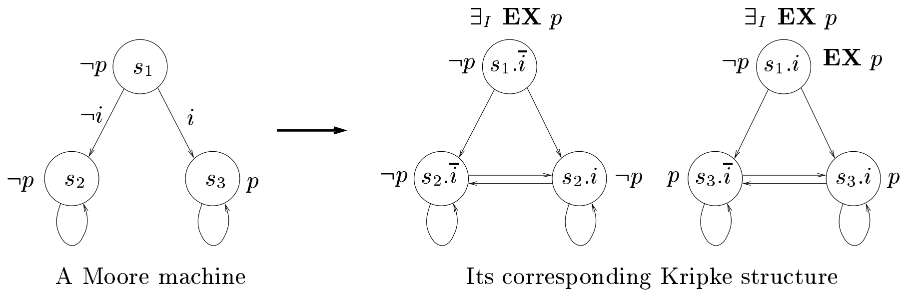

Fig. 3: An example showing the better expressiveness of iCTL

On the Kripke structure of figure 3, we can prove (by induction over its size) that any CTL formula won't see its truth value changed in $s_1.\bar{i}$, $s_2.\bar{i}$ and $s_2.i$ if we change the labelling of s_3. But the iCTL formula "$\exists_I$ **EX** p" distinguishes both cases. Since all CTL formulae are in iCTL, we have that iCTL is more expressive than CTL (for Kripke structures coming from definition 3).

4.4 *i*CTL and Other Logics

Modal μ-calculus is a logic dealing with labelled transition systems (thus, able to handle Moore machines), which contains the $\langle * \rangle$ and $[*]$ operators. $\langle * \rangle\ p$ is true in a state s if p is true in at least one of its successor, reachable by any transition. $[*]\ p$ is true in a state s if p is true in all the successors of s, reachable by any transition. We think that $\langle * \rangle$ in the μ-calculus is equivalent to $\exists_I$ **EX** in iCTL and that $[*]$ is equivalent to $\forall_I$ **AX**. Formulae $\exists_I$ **AX** p or $\forall_I$ **EX** p are in iCTL and have a meaning over Kripke structures obtained from Mealy machines. We didn't find equivalent formulae to those in the μ-calculus.

LTL does not present the same ambiguities than CTL since it only captures a set of infinite sequences and the sets of sequences of the Moore machine and of the derived Kripke structure are equivalent. So, something like "iLTL" would be useless.

5 Conclusion

The paper discusses the consequences of placing input configurations labelling transitions in Moore machines into the source states in the derived Kripke structure built to perform CTL model checking. This translation has an impact on the verification since a given CTL formula believed to be true or false on the Moore machine can have a different truth value on the obtained Kripke structure. This is due to the lack of expressiveness of CTL that does not take into account labelled transitions, as we find in Moore machines. To overcome this ambiguity, we introduce two operators, $\exists_I$ and $\forall_I$. We show that the obtained logic, named iCTL, is more expressive than CTL. We have implemented these operators in our model checker and it is our intention to verify complex systems with this logic.

References

1. R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. -T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy and T. Villa, *VIS: a System for Verification and Synthesis,* Proceedings of the Eighth International Conference on Computer Aided Verification CAV, pp. 428–432, 1996.
2. Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications,* ACM Trans. on Programming Languages and Systems, 8(2):244–263, 1986.
3. E. M. Clarke, D. E. Long, and K. L. McMillan, *Compositional Model Checking,* Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science, pp. 353–362, 1989.
4. E.A. Emerson, *Temporal and modal logic,* in Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, ed., J. van Leeuwen, chapter 16, pp. 995–1072, Elsevier, 1990.
5. K. L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.