A Context Dependent Equivalence Relation Between Kripke Structures

(Extended abstract)

Bernhard Josko
Computer Science Department, University of Oldenburg
2900 Oldenburg, Federal Republic of Germany

Abstract
In [BCG87] Browne, Clarke and Grumberg define a bisimulation relation on Kripke structure and give a characterization of this equivalence relation in temporal logic. We will generalize their results to reactive systems, which are modelled by Kripke structures together with some constraints describing some requirements how the environment has to interact with the module. Our results subsume the result of [BCM87] by using the constraint **true**. Furthermore it answers the questions raised in that paper how the equivalence of Kripke structures with fairness constraints can be characterized.

Keywords: temporal logic, Kripke structures, bisimulation, modular specification, reactive systems, hierarchical design

1 Introduction

In a top down design step of a large system one component may be splitted in several subcomponents. Not only the tasks of the subcomponents have to be specified but also the interface between the subcomponents have to be defined. On the one hand the interface has to declare the interconnections of the subcomponents i.e. the inputs and outputs of the components, and on the other hand the protocols for the exchange of data have to be specified too. E.g. if a subsystem consists of two components which are coupled asynchronously together (Fig. 1), we can use a 4-cycle signalling protocol to send data from one component to the other (cf. Fig. 2). Module M_1 is responsible for the Req signal and it has to guarantee that this signal is set and reset according to the given protocol, and M_2 is responsible for the signal Ack. Proceeding in the design process we may define a more concrete representation of module M_2 by splitting this component in several subcomponents together with interface

specifications or by defining an implementation. In such a design step we can use the fact that the environment (i.e. module M_1) acts according to the protocol and hence, it is only required that the module M_2 behaves correctly provided the environment guarantees the given interface constraint. Therefore a module specification is given by a pair (assm, spec) where assm describes some constraints on the environment and spec is the specification of the module, which has to be satisfied by the module provided the environment guarantees assm.

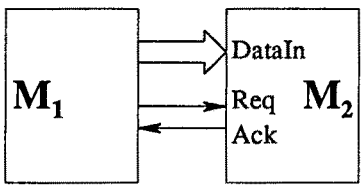

Fig. 1

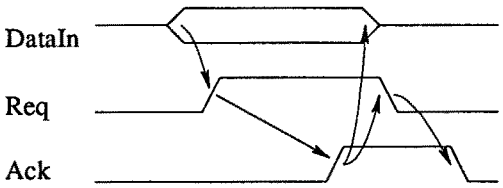

Fig. 2: 4-cycle signalling protocol

In this paper we investigate system design in the framework of temporal logic. We will use the branching time temporal logic CTL* [EH86] as specification language. In the context of temporal logics an implementation of a system is modelled by a state/transition-graph, also called a Kripke structure. A temporal logic formula is then interpreted in the associated computation tree, which is obtained by unravelling the transition graph. As we are considering open systems (or reactive systems [Pn85]), a module will be modelled by a Kripke structure K together with an interface constraint assm. The constraint assm restricts the possible paths in the computation tree. Hence to check the validity of a specification spec only those paths are considered which satisfy the given assumption assm. Considering the example in Fig. 1 together with Fig. 2 the environment constraints used by module M_2 can be defined in temporal logic as follows:

$\Box(Ack \wedge \neg Req \rightarrow [\neg Req \text{ unless } \neg Ack \wedge \neg Req])$

$\Box(Req \rightarrow [\text{stable(DataIn) unless } Ack])$

$\Box(Req \rightarrow [Req \text{ unless } Ack \wedge Req])$

$\Box(Ack \wedge Req \rightarrow \Diamond\neg Req)$

Fig. 3 shows a state/transition graph for M_2 (only the protocol is implemented).

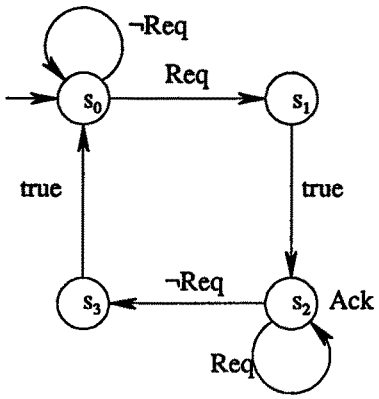

Fig. 3: Kripke structure K_1

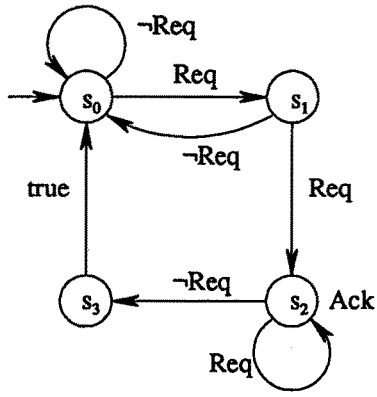

Fig. 4: Kripke structure K_2

During the design process a module implementation may be replaced by a new one. If a replaced module has already been verified to be correct w.r.t. its specification and one has derived properties of a composed module which contains that specific module, one is interested in the question whether the derived properties remain valid. Hence one is interested in an equivalence notion on Kripke structures, which guarantees that the replacement of a submodule by an equivalent one does not violate the correctness of derived properties of a composed (sub)system.

In Fig. 4 another implementation for module M_2 is given. In this implementation the request signal is checked after reading the input data. If the request signal is low - this is an violation of the protocol - the input data is ignored. In an environment which guarantees a correct behaviour according to the protocol, this step will never occurs. Thus both implementations, K_1 and K_2, are equivalent with respect to an environment satisfying the constraints.

Two structures will be called equivalent if they cannot be distinguished by any formula. In [BCG87] a bisimulation ([Mi83], [Pa81]) for **closed** systems, i.e. Kripke structures without constraints on the environment, is given. Two states s and s' are called bisimulation equivalent if they are labelled with the same atomic propositions and for every transition $s \rightarrow s_1$ in one structure there is a corresponding transition in the other structure leading to a state s'_1 which is equivalent to s_1. It is shown that this relation can be characterized by the fact that both structures satisfy the same temporal logic formulae. In this paper we will show how the results of [BCG87] can be generalized to **open** systems. We will relativize the correspondence of

equivalent steps by requiring that only for those transitions which are on a path satisfying the environment constraints there are corresponding transitions in the other module. We will show that this bisimulation with respect to an assumption assm characterizes the structures satisfying the same set of CTL* formulae relatively to the given assumption assm. i.e. two structures K_1 and K_2 are bisimulation equivalent w.r.t. assm iff the modules (K_1,assm) and (K_2,assm) are not distinguishable by any CTL* formula.

For action based transition systems like CCS Larsen has given a notion of bisimulation relatively to some contexts [La86], [LM87], but there is no relation to temporal logic specifications.

2 Basic definitions

A *Kripke structure* is given by $K = (S, R, s^0, L)$, where S is a finite set of states, $s^0 \in S$ is the initial state and R is a transition relation $R \subseteq S \times S$, such that for every state s there is some state s' with $(s, s') \in R$, and L is a labelling function which associates with every state $s \in S$ a set of atomic propositions, which are valid in that state.

Dealing with reactive systems one should use labelled transitions, as a transition depends on the input signals. A *Kripke structure with inputs* is given by $K = (S, R, s^0, IN, OUT, L)$ where S is a set of states, s^0 is the initial state, IN and OUT are disjoint set of propositions, L is a labelling of states with subsets of OUT, and R is a transition relation with $R \subseteq S \times BExpr(IN) \times S$, such that for every satisfiable boolean expression b and every state s there is some state s' with $(s, b, s') \in R$. IN is a set of propositions on the input signals (usually the input signals themselves) and OUT are propositions on the internal and output signals. The transition labels are constraints on the input signals, restricting a transition to the instances where the actual input signals satisfy the given boolean expression. For every Kripke structure K with inputs there is a corresponding Kripke structure K' (without inputs) [Br86]. Therefore we will use the usual notion of Kripke structure in this paper. Fig. 5 resp. Fig. 6 shows the transformed Kripke structures of Fig. 3 resp. Fig. 4.

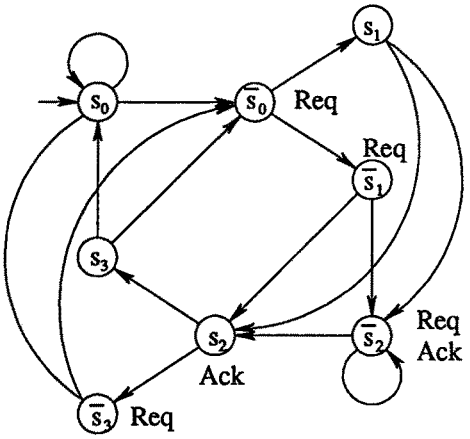

Fig. 5: Kripke structure K_1 transformed

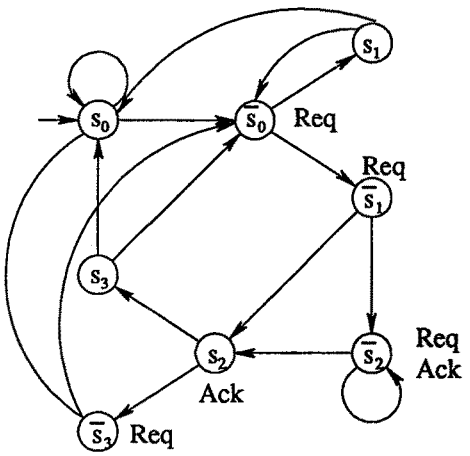

Fig. 6: Kripke structure K_2 transformed

Given a Kripke structure $K = (S, R, s^0, L)$, $\pi = (s_i \mid i \in \mathbb{N})$ is a *path* in K if $(s_i, s_{i+1}) \in R$ for all $i \in \mathbb{N}$. We will refer to the i-th state s_i also by $\pi(i)$. The suffix of a path starting at state $\pi(i)$ will be denoted by π^i. The (infinite) *computation tree* of K is obtained by unravelling the Kripke structure (considered as a graph) starting with the initial state s^0.

PTL [Pn77] is the linear propositional temporal logic defined by

$$f ::= a \mid \neg f \mid f_1 \wedge f_2 \mid f_1 \vee f_2 \mid Xf \mid [f_1 \, U \, f_2] \mid \Box f \mid \Diamond f,$$

where a denotes an atomic proposition. $[f_1$ unless $f_2]$ will be used as an abbreviation for $\Box f_2 \vee [f_1 \, U \, f_2]$. The validity of a PTL formula f along a path π will be denoted by $\pi \models f$. We also call a path π *f-good* if π satisfies the formula f. A state s is called *f-good* iff there is some path π starting at s with is f-good.

CTL and CTL* are branching time temporal logics which are defined by the following rules:

(1) $sf ::= a \mid \neg sf \mid sf_1 \wedge sf_2 \mid sf_1 \vee sf_2 \mid \forall pf \mid \exists pf$

(2) $pf ::= sf \mid \neg pf \mid pf_1 \wedge pf_2 \mid pf_1 \vee pf_2 \mid Xpf \mid [pf_1 \, U \, pf_2]$

(3) $pf ::= Xsf \mid [sf_1 \, U \, sf_2]$

The rules in (1) describe the building of state formulae and the rules in (2) and (3) describe the construction of path formulae. CTL is the set of all state formulae defined by (1) and (3) and CTL* is the set of all state formulae defined by (1) and (2). Furthermore we will use the path formula $\Diamond f$ as an abbreviation for [**true** U f], and $\Box f$ as an abbreviation for $\neg\Diamond\neg f$. The

semantics of CTL and CTL* for closed systems is as usual (cf. [EH86]).

A module is given by a pair (K,assm), where K is a Kripke structure and assm is a PTL formula. The interpretation of a CTL* formula spec in a module (K,assm) can be described as follows:

(1) Construct the computation tree with root s^0.

(2) Mark all paths starting at s^0 which satisfy the constraint assm.

(3) Interpret the formula spec as in the usual interpretation of CTL* formulae but with the restriction that all path quantifiers are restricted to the marked paths.

To give an inductive definition of the semantics we need a description of the behaviour of the environment at a specific instance on a computation path. The expected behaviour of the environment at instance i on a path π is determined by the given interface constraint assm and the computation history along π. E.g. if $\Box$(Req $\rightarrow$ [Req unless Req $\wedge$ Ack]) is a conjunctive of the interface constraint and assume that the signal Req is set to high at instance j (i.e. the proposition Req is valid at instance j) then [Req unless Req $\wedge$ Ack] is also a part of the expected behaviour of the environment at the next time instance. Using the computation history along a path π we can define a formula $assm_i(\pi,assm)$ which describes the expected behaviour of the environment at instance i on the computation path π.

Given a PTL formula assm and a state s we can define a PTL formula next-assm(assm,s) with the following property:

$\pi \models assm$ iff $\pi^1 \models$ next-assm(assm,s) for any path π starting at s.

This defintion can be extended to all time instances along a path. Given a path π and a formula assm, we can determine for every $i \in \mathbf{N}$ a formula $assm_i$ which is valid on π^i iff π satisfies assm. We define

$assm_0(f,\pi) := f$ $\qquad$ $assm_{k+1}(f,\pi) := \text{next-assm}(assm_k(f,\pi), \pi(k))$

Now we can give an inductive definition of the validity of a CTL* formula in a module (K,assm) using the notion of next-assm(assm,s) and $assm_i(assm,\pi)$. Given a Kripke structure K and a context assm, the validity of a formula f at a state s, denoted by (K, assm, s) $\models$ f, is defined as follows:

(K, assm, s) $\models \forall$pf iff for all assm-good paths π starting at s: (K, assm, π) $\models$ pf

(K, assm, s) $\models \exists$pf iff there is some assm-good path π starting at s with

$(K, assm, \pi) \models pf$

$(K, assm, \pi) \models sf$ iff $(K, assm, \pi(0)) \models sf$

$(K, assm, \pi) \models Xpf$ iff $(K, assm_1(assm,\pi), \pi^1) \models pf$

$(K, assm, \pi) \models [pf_1 \; U \; pf_2]$ iff there is some k with $(K, assm_k(assm,\pi), \pi^k) \models pf_2$ and for all j, $0 \leq j < k$: $(K, assm_j(assm,\pi), \pi^j) \models pf_1$

The other cases are straightforward. Furthermore, we say that a module (K, assm) satisfies a specification f, denoted by $(K, assm) \models f$, iff $(K, assm, s^0) \models f$.

3 Relativized Equivalence of Kripke Structure

Usually two Kripke structures K and K' are equivalent if both satisfy the same set of formulae. Dealing with modules embedded in an environment which has to guarantee a behaviour according to an interface specification assm, we use a relativized equivalence relation, which demands only that both modules should have the same behaviour in such an environment, i.e. (K,assm) and (K',assm) cannot be distinguished by any formula.

Two Kripke structures K and K' are *equivalent relatively to assm* $\in$ PTL, denoted by $K \equiv_{assm} K'$, iff for all CTL* formulae f: $(K, assm) \models f$ iff $(K', assm) \models f$.

This (semantical) equivalence relation can be defined by a syntactical relation on the structures (bisimulation) as we will show in the sequel. Furthermore we will characterize the equivalence class of a given Kripke structure by a CTL formula. We will define the relativized bisimulation by a chain of approximations considering paths up to depth i.

Given two Kripke structures $K = (S,s^0,R,L)$ and $K' = (S',s'^0,R',L')$ with the same set ATOM of atomic propositions and given a PTL formula assm we define the relativized equivalence relations $BISIM_i(assm) \subseteq S \times S'$ as follows:

(1) $(s,s') \in BISIM_0(assm)$ iff $L(s) = L'(s')$

(2) $(s,s') \in BISIM_{i+1}(assm)$ iff

- (a) $L(s) = L'(s')$ and
- (b) for every next-assm(assm,s) good successor s_1 of s there is a successor s'_1 of s' with $(s_1,s'_1) \in BISIM_i(next\text{-}assm(assm,s))$
- (c) for every next-assm(assm,s') good successor s'_1 of s' there is a successor s_1 of s with $(s_1,s'_1) \in BISIM_i(next\text{-}assm(assm,s))$

(3) $(s,s') \in BISIM(assm)$ iff $(s,s') \in BISIM_i(assm)$ for all $i \in \mathbb{N}$.

(4) K BISIM(assm) K' iff $(s^0,s'^0) \in$ BISIM(assm).

Example

Consider the Kripke structures of Fig. 5 and Fig. 6. Without consideration of the environment constraints the two structures are different, as K_1 can proceed from state s_1 to $\overline{s}_2$ or s_2, whereas in K_2 the next states of s_1 are s_0 or $\overline{s}_0$ as the input signal Req is low in state s_1. But as the environment has to guarantee that the signal Req remains high unless the response Ack occurs - this is expressed by the assumption assm = $\Box$(Req $\rightarrow$ [Req unless Req $\wedge$ Ack]) -, both structures are equivalent, as under the constraint assm the state s_1 in K_1 and the state s_1 in K_2 are not reachable.

The bisimulation BISIM(assm) and the equivalence relation $\equiv_{assm}$ coincide. To prove this fact we first show that two Kripke structures are equivalent relatively to a given constraint assm if they are bisimulation equivalent.

Theorem 1

If (s,s') $\in$ BISIM(assm) then for all f $\in$ CTL*: ((K,assm,s) $\models$ f iff (K',assm,s') $\models$ f).

For the reverse direction we give a characterization of the bisimulation class of a Kripke structure by a CTL formula. Basicly this formula describes the computation tree of the structure. Two states which are bisimulation equivalent have corresponding computation trees w.r.t. the given constraint on the environment. For a state s, let $CT_n(s)$ denote the computation tree of depth n rooted at s. We can describe the computation tree $CT_n(s)$ w.r.t. an assumption assm by a CTL formula. The formula $F_{CT,n}(s,assm)$ is defined by:

$F_{CT,0}(s,assm) = a_1 \wedge ... a_n \wedge \neg b_1 \wedge ... \wedge \neg b_m$,

where $L(s) = \{a_1, ..., a_n\}$ and $ATOM \setminus L(s) = \{b_1, ..., b_m\}$

$F_{CT,k+1}(s,assm) = F_{CT,0}(s,assm) \wedge$

$\bigwedge\{ \exists X\, F_{CT,k}(s',next\text{-}assm(assm,s)) \mid s' \text{ is a next-assm(assm,s)-good successor of } s \} \wedge$

$\forall X\, (\bigvee\{ F_{CT,k}(s',next\text{-}assm(assm,s)) \mid s' \text{ is a next-assm(assm,s)-good successor of } s\})$

As there is a finite depth m such that $CT_m(s^0)$ w.r.t. assm characterizes the (infinite) computation tree we obtain:

Theorem 2

Given a Kripke structure K with initial state s^0 and an assumption assm, then there is a CTL formula $F_{BISIM}(K,assm)$ that characterizes that structure up to BISIM(assm)-equivalence.

Combining the results of Theorem 1 and Theorem 2 we obtain the following characterizations.

Corollary 1

(1) Given two structures K and K' then it holds:
$(K,K') \in BISIM(assm)$ iff $(\forall f \in CTL^* : (K,assm) \models f$ iff $(K',assm) \models f)$

(2) Given two structures K and K' then it holds:
$(K,K') \in BISIM(assm)$ iff $(\forall f \in CTL : (K,assm) \models f$ iff $(K',assm) \models f)$

(3) Given two structures K and K' then it holds:
If there is some CTL* formula f with $(K,assm) \models f$ but $(K',assm) \models \neg f$, then there is already a CTL formula f' which distinguishes both structures.

4 Conclusion

In this paper we have defined a relativized bisimulation between Kripke structures and we have characterized this relation in temporal logic. Our equivalence relation is a generalization of the relation given in [BCG87], the results of [BCG87] are subsumed by our results by using the assumption **true**. Our bisimulation is a strong bisimulation, but we can also define a weak bisimulation by weakening the condition of a corresponding step: for every step $s \rightarrow s_1$ on an assm-good path there should be a corresponding finite path in the other structure. To give a temporal characterization of the weak bisimulation we have to omit the next operator. This leads to a generalization of the corresponding result of [BCG87]. As fairness constraints may be specified as an assumption, we can characterize the equivalence of Kripke structures with fairness constraints, too. This solves a problem raised in [BCG87].

If a temporal specification of a system composed of several modules is derived from the specifications of the modules, and the modules are proved correct w.r.t. their specifications, one can replace every module by an equivalent one without loosing the correctness of the derived specification. This can be done as two equivalent modules can not be distinguished by any temporal formula.

Furthermore the equivalence relation is decidable, as the formula $F_{BISIM}(K,assm)$ is constructable and the validity of a formula spec in a module (K',assm) is decidable.

If the given constraints on the environment are safety properties, the definition of an assm-good state can be weakened to the requirement that the state has to satisfies only the disjunction of all sets literal(C), where C is a closure of assm. Hence this condition can be checked locally.

In the context of the computer architecture design language AADL we use a temporal logic MCTL for module specification [DD89], [DDGJ89], [Jo89]. MCTL consists of pairs (assm, spec) where assm is a restricted PTL formula and spec is a CTL formula. By this paper we have an appropriate notion of equivalence for that logic, which can be used in the design process within AADL.

5 References

[BCG87] M.C. Browne, E.M. Clarke, O. Grumberg: Charactering Kripke Structures in Temporal Logic. Tech. Report CMU-CS-87-104, Carnegie Mellon University, Pittsburgh (1987)

[Br86] M. Browne: An improved algorithm for the automatic verification of finite state systems using temporal logic. Symp. Logics in Computer Science, pp. 260 - 266, 1986

[DD89] W. Damm, G. Döhmen: AADL: A net based specification method for computer architecture design. in: de Bakker (Ed.): Languages for Parallel Architectures: Design, Semantics, and Implementation Models. Wiley & Sons, 1989

[DDGJ90] W. Damm, G. Döhmen, V. Gerstner, B. Josko: Modular verification of Petri nets: The temporal logic approach. REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, LNCS 430, pp. 180 - 207, 1990

[EH86] E.A. Emerson, J.Y. Halpern: "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. Journal of the ACM 33, pp. 151-178, 1986

[Jo90] B. Josko: Verifying the correctness of AADL modules using model checking. REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, LNCS 430, pp. 386 - 400, 1990

[Kr87] F. Kröger: Temporal Logic of Programs. EATCS-Monographs, Springer, 1987

[La86] K.G. Larsen: Context-dependent bisimulation between processes. Ph.D. Thesis, Edingburgh, 1986

[LM87] K.G. Larsen, R. Milner: Verifying a protocol using relativized bisimulation. ICALP 87, LNCS 267, pp. 126-135, 1987

[LP85] O. Lichtenstein, A. Pnueli: Checking that finite state concurrent programs satisfy their linear specification. 12th ACM Symp. on Principles of Programming Languages, pp. 97-107, 1985

[Mi83] R. Milner: Calculi for synchrony and asynchrony. TCS 25, 1983

[Pa81] D. Park: Concurrency and automata on infinite sequences. LNCS 104, pp167-183, 1981

[Pn77] A. Pnueli: The temporal logic of programs. 18th Annual Symposium on Foundations of Computer Science, 1977

[Pn85] A. Pnueli: Linear and branching structures in the semantics and logics of reactive systems. ICALP 85, LNCS 194, 1985