

Consiglio Nazionale delle Ricerche

**ISTITUTO DI ELABORAZIONE
DELLA INFORMAZIONE**

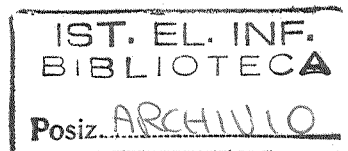
PISA

Model Checking for Action-based Logics




A. Fantechi, S. Gnesi, G. Ristori



Nota Interna B4-41

Ottobre 1992



Model Checking for Action-based Logics

Alessandro Fantechi  Stefania Gnesi  Gioia Ristori 

 Istituto di Elaborazione dell'Informazione - C.N.R. Pisa.
 Università degli Studi di Pisa.

A model checker is described which supports proving logical properties of concurrent systems. The logical properties can be described in different action based logics (variants of Hennessy-Milner Logic). The tool is based on the EMC model checker for the logic CTL. It employs therefore a set of translation functions from the considered logics to CTL, as well as a model translation function from Labelled Transition Systems (models of the action-based logics) to Kripke Structures (models for CTL). The obtained tool performs model checking in linear time complexity, and its correctness is guaranteed by the proof that the set of translation functions, coupled with the model translation function, preserve satisfiability of logical formulae.

1 Introduction

Logic is a good candidate to provide abstract specifications of concurrent systems; different types of temporal and modal logics have been proposed as suitable for specifying system properties [EH86, HM85, MP89] due to their ability to deal with the notions of *necessity*, *possibility*, *eventuality*, etc.. Logics have been equipped with model checkers to prove satisfiability of formulae and thus system properties: a system, usually a finite state system, is considered as a potential model for the formula expressing the desired property.

Actually, very interesting temporal logics like CTL and CTL*, interpreted on Kripke Structures, which require formulating properties of systems in terms of their states, have been put forward [BCG88, EH86, ES89]; for CTL a sophisticated and linear time model checker has also been developed [CES86].

Due to the success of process algebras, other logics have been proposed, which are interpreted over Labelled Transition Systems [HM85, BGS88, Sti90, DV90b, DV90a, Lar90]. Kripke Structures and Labelled Transition Systems differ mainly because in the former states are labelled to describe how they are modified by the transitions, while in the latter transitions are labelled to describe the actions which cause state changes.

The definition of logics in the setting of Labelled Transition Systems has paid attention to the key concepts of process algebras, namely combinators for transition systems and behavioural equivalences. These "action" based logics differ in expressive power, that is in the classes of properties definable by them, and also in their adequacy [HM85] with respect to behavioural equivalences.

In order to verify properties of concurrent systems defined by means of process algebras, one or the other of these logics can be used accordingly to the problem to be solved and to the experience

of the verifier. The verification phase would take advantage in using automatic tools, therefore efficient model checker for them would be necessary.

Different ways could have been followed to develop model checkers for these logics:

- 1) define for each logic a proper model checker;
- 2) reuse existing model-checkers and in particular:
 - 2.1) using a μ -calculus model-checker [CPS90, CS91], of which these logics are subsets;
 - 2.2) using the EMC model checker for the logic CTL [CES86]; in fact the action-based logics, without fixed point operator, are subsets of CTL, modulo a translation between the underlying models.

Basing on the experience drawn from the construction of the verification environment presented in [DFGR92], we have chosen the last alternative to realize a model checker which allows to reason on several variants of HML, ranging from the original definition to ACTL (HML [HM85], HML' [Sti90], HML_{LJ} [DV90b] and ACTL [DV90a]).

The model checker we propose is hence based on the EMC model checker, and it employs two modules performing the translations between the variants of HML and CTL and a translation between Labelled Transition Systems and Kripke Structures.

The choice of EMC has been due both to the fact that it is an already tested and widely used tool and to the fact that it permits a linear time model checking; since the model translation functions have linear complexity, we achieve linear model checking also for the variants of HML and linear complexity is the best result that can be expected.

2 Logics for Labelled Transition Systems

In this Section we present the logics HML, HML', HML_{LJ} and ACTL, whose interpretation domains are Labelled Transition Systems.

A *Labelled Transition System* (or *LTS*) is a 4-tuple $\mathcal{A} = (Q, A \cup \{\tau\}, \rightarrow, 0_Q)$ where:

- Q is a set of *states*;
- A is a finite, non-empty set of *visible actions*; the *silent action* τ is not in A ;
- $\rightarrow \subseteq Q \times (A \cup \{\tau\}) \times Q$ is the *transition relation*; an element $(r, \alpha, q) \in \rightarrow$ is called a *transition*, and is written as $r \xrightarrow{\alpha} q$;
- 0_Q is the initial state.

We let $A_\tau = A \cup \{\tau\}$; $A_\varepsilon = A \cup \{\varepsilon\}$, $\varepsilon \notin A_\tau$. Moreover, we let $r, q, s \dots$ range over states; a, b, \dots over A ; α, β, \dots over A_τ and k, \dots over A_ε .

Paths over a LTS, $\mathcal{A} = (Q, A_\tau, \rightarrow, 0_Q)$, are now introduced; we let π, ρ, σ, η range over paths.

- A finite or infinite sequence $(q_0, \alpha_0, q_1) (q_1, \alpha_1, q_2) \dots$ is called a *path* from q_0 ; a path that cannot be extended, i.e. is infinite or ends in a state without outgoing transitions, is called a *fullpath*; the *empty path* consists of a single state $q \in Q$ and it is denoted by q ;
- if $\pi = (q_0, \alpha_0, q_1) (q_1, \alpha_1, q_2) \dots$ we denote the starting state, q_0 , of the sequence by *first*(π) and the last state in the sequence (if the sequence is finite) by *last*(π); if π is an empty path (i.e. $\pi = q$), then *first*(π) = *last*(π) = q
- concatenation of paths is denoted by juxtaposition: $\pi = \rho\theta$; it is only defined if ρ is finite and *last*(ρ) = *first*(θ).

When $\pi = \rho\theta$ we say that θ is a *suffix* of π and that it is a *proper suffix* if $\rho \neq q, q \in Q$.

We write $path(q)$ for the set of fullpaths from q .

On LTSs several equivalences can be defined; among them strong and weak bisimulation [Par81, Mil80, Mil83] and branching bisimulation [vGW89]; see Appendix A for a complete definition of these equivalence relations.

Given a logic \mathcal{J} , defined on LTSs, with a satisfaction relation $\vDash_{\mathcal{A}}$, written $q \vDash_{\mathcal{A}} \phi$ and read " q , state of \mathcal{A} , satisfies the property ϕ expressed in \mathcal{J} ", the definition of this relation induces an equivalence $\equiv_{\mathcal{J}}$ between states of \mathcal{A} which enjoy the same properties. Formally, we define:

$$q \equiv_{\mathcal{J}} q' \quad \text{iff} \quad \mathcal{F}(q) = \mathcal{F}(q')$$

where: $\mathcal{F}(q) = \{\psi : \psi \in \mathcal{J} \wedge q \vDash_{\mathcal{A}} \psi\}$

Now, if \equiv is an equivalence relation between LTSs, it is possible to relate \equiv and $\equiv_{\mathcal{J}}$ by means of the notion of adequacy: a logic \mathcal{J} is *adequate* [HM85] with respect to \equiv , if for every pair of states q and q' :

$$q \equiv_{\mathcal{J}} q' \quad \text{iff} \quad q \equiv q'.$$

Each language of logic formulae that we present in this Section differs for its adequacy with respect to a certain equivalence and for its expressive power in terms of the definable properties.

2.1 HML

HML (Hennessy Milner Logic) is a logic whose formulae are interpreted on LTSs [HM85]. HML formulae have the following syntax:

$$\phi ::= \text{true} \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle \phi \quad \text{where } a \in A$$

Given a LTS $\mathcal{A} = (Q, A \cup \{\tau\}, \rightarrow, 0_Q)$, the meaning of HML formulae is given defining a satisfaction relation $\vDash_{\mathcal{A}}$ between the set Q of states and the set of HML formulae; $q \vDash_{\mathcal{A}} \phi$ means that the state q satisfies the formula ϕ .

$$\begin{aligned} q \vDash_{\mathcal{A}} \text{true} & \\ q \vDash_{\mathcal{A}} \neg\phi & \quad \text{iff} \quad \text{not } q \vDash_{\mathcal{A}} \phi \\ q \vDash_{\mathcal{A}} \phi_1 \wedge \phi_2 & \quad \text{iff} \quad q \vDash_{\mathcal{A}} \phi_1 \text{ and } q \vDash_{\mathcal{A}} \phi_2 \\ q \vDash_{\mathcal{A}} \langle a \rangle \phi & \quad \text{iff} \quad \text{there exists } \rho \in path(q) \text{ such that } \rho = (q, a, q')\rho' \text{ and } q' \vDash_{\mathcal{A}} \phi \end{aligned}$$

We can define on the basis of the primitive operator $\langle a \rangle \phi$ the derived modality $[a] \phi$ as $\neg \langle a \rangle \neg \phi$. Informally $[a] \phi$ is satisfied by q if, whenever $q \xrightarrow{a} q'$, then q' satisfies ϕ .

In [HM85] it is shown that HML is adequate with respect to strong bisimulation equivalence.

2.2 HML'

In order to take into account a notion of observability on LTSs, a variant of HML can be given; we will call this new logic HML', whose abstract syntax is the following:

$$\phi ::= \text{true} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle a \rangle\rangle \phi \mid \langle\langle \varepsilon \rangle\rangle \phi$$

The satisfaction relation for the new modalities is the following:

$$q \vDash_{\mathcal{A}} \langle\langle a \rangle\rangle \phi \quad \text{iff} \quad \text{there exist } \rho \in \text{path}(q) \text{ and a decomposition } \eta\theta \text{ of } \rho \text{ such that } \theta = (t, a, t')\theta' \text{ and } t' \vDash_{\mathcal{A}} \phi, \text{ and all suffixes of } \eta \text{ of the form } \eta' = (s, \alpha, s')\eta'' \text{ are such that } \alpha = \tau.$$

$$q \vDash_{\mathcal{A}} \langle\langle \varepsilon \rangle\rangle \phi \quad \text{iff} \quad \text{there exist } \rho \in \text{path}(q) \text{ and a decomposition } \eta\theta \text{ of } \rho \text{ such that } \text{first}(\theta) \vDash_{\mathcal{A}} \phi, \text{ and all suffixes of } \eta \text{ of the form } \eta' = (s, \alpha, s')\eta'' \text{ are such that } \alpha = \tau.$$

We can define, as in the HML case, the derived modalities $[[\alpha]] \phi = \neg \langle\langle \alpha \rangle\rangle \neg \phi$.

Alternatively we could have defined the above modalities by using the double arrow relations defined by: $p = \varepsilon \Rightarrow q$ if $p \xrightarrow{\tau^*} q$ and $p = a \Rightarrow q$ if $p \xrightarrow{\varepsilon} p' \xrightarrow{a} q' = \varepsilon \Rightarrow q$. In fact, informally, $\langle\langle a \rangle\rangle \phi$ means that there exists a state q which satisfies ϕ , such that $p = a \Rightarrow q$, and $\langle\langle \varepsilon \rangle\rangle \phi$ means that there exists a state q which satisfies ϕ , such that $p = \varepsilon \Rightarrow q$.

HML' can be easily proved to be adequate with respect to observational equivalence [Sti90].

2.3 HML_U

A variant of HML', named HML_U, has been defined in [DV90b], which has indexed until operators instead of the indexed diamond modalities:

$$\phi ::= \text{true} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \langle \varepsilon \rangle \phi' \mid \phi \langle a \rangle \phi'$$

with the following satisfaction relation for the until operators:

$$q \vDash_{\mathcal{A}} \phi \langle a \rangle \phi' \quad \text{iff} \quad \text{there exist } \rho \in \text{path}(q) \text{ and a decomposition } \eta\theta \text{ of } \rho \text{ such that:} \\ \theta = (t, a, t')\theta' \text{ and } t \vDash_{\mathcal{A}} \phi \text{ and } t' \vDash_{\mathcal{A}} \phi'; \\ \text{all suffixes of } \eta \text{ of the form } \eta' = (s, \alpha, s')\eta'' \text{ are such that } s \vDash_{\mathcal{A}} \phi \text{ and } \alpha = \tau.$$

$$q \vDash_{\mathcal{A}} \phi \langle \varepsilon \rangle \phi' \quad \text{iff} \quad q \vDash_{\mathcal{A}} \phi' \text{ or there exist } \rho \in \text{path}(q) \text{ and a decomposition } \eta\theta \text{ of } \rho \text{ such that:} \\ \theta = (t, \tau, t')\theta' \text{ and } t \vDash_{\mathcal{A}} \phi \text{ and } t' \vDash_{\mathcal{A}} \phi'; \\ \text{all suffixes of } \eta \text{ of the form } \eta' = (s, \alpha, s')\eta'' \text{ are such that } s \vDash_{\mathcal{A}} \phi \text{ and } \alpha = \tau.$$

In [DV90b] it is shown that HML_U is adequate with respect to branching bisimulation equivalence.

2.4 ACTL

ACTL is an action based version of the CTL logic. In order to define the logic ACTL [DV90a], an auxiliary logic of actions is introduced.

The collection A for *action formulae* over A is defined by the following grammar where χ, χ' , range over action formulae, and $a \in A$:

$$\chi ::= a \mid \neg\chi \mid \chi \wedge \chi'$$

The satisfaction of an action formula χ by an action a , notation $a \models \chi$, is defined inductively by:

$$\begin{aligned} a \models b & \quad \text{iff} \quad a = b; \\ a \models \neg\chi & \quad \text{iff} \quad a \not\models \chi; \\ a \models \chi \wedge \chi' & \quad \text{iff} \quad a \models \chi \text{ and } a \models \chi'. \end{aligned}$$

The syntax of the logic ACTL, a subset of ACTL* [DV90a], is defined by the state formulae generated by the following grammar, where ϕ, ϕ', \dots range over state-formulae, γ over path formulae and χ and χ' are action formulae:

$$\begin{aligned} \phi & ::= \text{true} \mid \neg\phi \mid \phi \wedge \phi' \mid \exists\gamma \mid \forall\gamma \\ \gamma & ::= X_\chi\phi \mid X_\tau\phi \mid \phi \chi U_{\chi'}\phi' \mid \phi \chi U\phi' \end{aligned}$$

We give below the satisfaction relation for ACTL formulae.

Satisfaction of an ACTL-formula ϕ (γ) by a state q (path ρ), notation $q \models_{\mathcal{A}} \phi$ ($\rho \models_{\mathcal{A}} \gamma$), is given inductively by:

- $q \models_{\mathcal{A}} \text{true}$ always;
- $q \models_{\mathcal{A}} \neg\phi$ iff $q \not\models_{\mathcal{A}} \phi$;
- $q \models_{\mathcal{A}} \phi \wedge \phi'$ iff $q \models_{\mathcal{A}} \phi$ and $q \models_{\mathcal{A}} \phi'$;
- $q \models_{\mathcal{A}} \exists\gamma$ iff there exists a path $\theta \in \text{path}(q)$ such that $\theta \models_{\mathcal{A}} \gamma$;
- $q \models_{\mathcal{A}} \forall\gamma$ iff for all paths $\theta \in \text{path}(q)$ $\theta \models_{\mathcal{A}} \gamma$;
- $\rho \models_{\mathcal{A}} \phi \chi U_{\chi'}\phi'$ iff there exists a suffix θ of ρ , $\theta = (q, a, q')\theta'$, s.t. $q' \models_{\mathcal{A}} \phi'$, $a \models \chi'$, $q \models_{\mathcal{A}} \phi$ and for all $\eta = (r, \beta, r')\eta'$, suffixes of ρ , that have θ as proper suffix, we have $r \models_{\mathcal{A}} \phi$ and ($\beta \models \chi$ or $\beta = \tau$);
- $\rho \models_{\mathcal{A}} \phi \chi U\phi'$ iff there exists a suffix θ of ρ s. t. $\text{first}(\theta) \models_{\mathcal{A}} \phi'$ and for all $\eta = (r, \beta, r')\eta'$ of which θ is a proper suffix we have $r \models_{\mathcal{A}} \phi$ and ($\beta \models \chi$ or $\beta = \tau$);
- $\rho \models_{\mathcal{A}} X_\chi\phi$ iff $\rho = (q, a, q')\theta$ and $q' \models_{\mathcal{A}} \phi$ and $a \models \chi$;
- $\rho \models_{\mathcal{A}} X_\tau\phi$ iff $\rho = (q, \tau, q')\theta$ and $q' \models_{\mathcal{A}} \phi$.

The indexed next modalities $X_\chi\phi$, $X_\tau\phi$ say that in the next state of the path, reached respectively with an action in χ or with a τ , the formula ϕ holds; the meaning of the indexed until modalities introduced above is better clarified in Fig. 1:

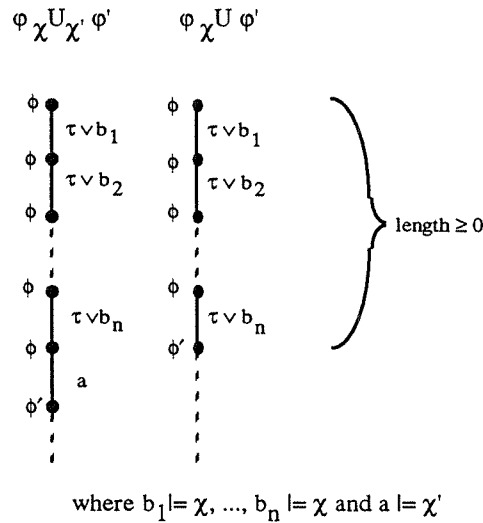


Fig. 1 The meaning of the indexed until modalities.

ACTL is adequate with respect to strong bisimulation and ACTL- $\{X\}$ is adequate w.r.t. branching bisimulation [DV90a].

3 CTL: a Logic for Kripke Structures

CTL [CES86] is a language of state formulae interpreted over Kripke Structures, and it is just a subset of CTL* [EH86]; CTL* combines linear and branching time operators; its syntax is given in terms of path formulae that are interpreted over computation paths and state formulae that are true or false of a state. CTL is the branching time subset of CTL* in which every linear time operator is immediately preceded by a path quantifier.

A *Kripke Structure* (or *KS*) is a 5-tuple $\mathcal{K} = (S, AP, L, \rightarrow, 0_S)$ where:

- S is a set of *states*;
- AP is a finite, non-empty set of *atomic proposition names* ranged over by p, p_1, \dots ;
- $L: S \rightarrow 2^{AP}$ is a function that labels each state with a set of atomic propositions true in that state;
- $\rightarrow \subseteq S \times S$ is the *transition relation*; an element $(r, s) \in \rightarrow$ is called a *transition* and it is written as $r \rightarrow s$;
- 0_S is the *initial state*.

The notation for paths of Labelled Transition Systems carries over to Kripke Structures in the obvious way. The only difference is that transitions are no longer triples but pairs, so a path of a Kripke Structure is a sequence of pairs $(s_0, s_1) (s_1, s_2) \dots$.

CTL is defined as the set of state formulae ϕ given by the following grammar, where γ ranges on path formulae and P ranges on the atomic predicates in AP :

$$\phi ::= P \mid \neg \phi \mid \phi \wedge \phi \mid \mathbf{E}\gamma \mid \mathbf{A}\gamma$$

$$\gamma ::= \mathcal{X} \phi \mid \phi \mathcal{U} \phi$$

Let $\mathcal{K} = (S, AP, L, \rightarrow, 0_S)$ be a Kripke Structure. *Satisfaction* of a state-formula ϕ (path-formula γ) by a state s (a maximal path π), denoted by $s \models_{\mathcal{K}} \phi$ and $\pi \models_{\mathcal{K}} \gamma$ respectively, is defined by:

State-formulae

$s \models_{\mathcal{K}} P$	iff	$P \in L(s)$ ($P \in AP$)
$s \models_{\mathcal{K}} \neg \phi$	iff	not ($s \models_{\mathcal{K}} \phi$)
$s \models_{\mathcal{K}} \phi \wedge \phi'$	iff	($s \models_{\mathcal{K}} \phi$) and ($s \models_{\mathcal{K}} \phi'$)
$s \models_{\mathcal{K}} \mathcal{E} \gamma$	iff	for some π such that $\text{first}(\pi) = s$: $\pi \models_{\mathcal{K}} \gamma$
$s \models_{\mathcal{K}} \mathcal{A} \gamma$	iff	for all π such that $\text{first}(\pi) = s$: $\pi \models_{\mathcal{K}} \gamma$

Path-formulae

$\pi \models_{\mathcal{K}} \mathcal{X} \phi$	iff	$ \pi \geq 1$ and $\text{first}(\pi^1) \models_{\mathcal{K}} \phi$
$\pi \models_{\mathcal{K}} \phi \mathcal{U} \phi'$	iff	there exists $i \geq 0$ such that $\text{first}(\pi^i) \models_{\mathcal{K}} \phi'$ and for every $j \geq 0$: $j < i$ implies $\text{first}(\pi^j) \models_{\mathcal{K}} \phi$

On Kripke Structures several equivalences can be defined; among these, the definition of strong bisimulation and stuttering equivalence given in [BCG88], where it has been proved that CTL is adequate with respect to strong bisimulation and $\text{CTL}-\{\mathcal{X}\}$ is adequate w.r.t. stuttering equivalence on Kripke Structures.

4 From LTS to KS

In order to compare action-based logics and CTL and to translate the ones into the other, we need to give a relation between Labelled Transition Systems and Kripke Structures. In the literature, we find two different definitions of such a relation: the first one [JKP90], called in the following T_S , is based on a "strong" translation of LTSs into KSs, i.e. any action in the LTS is considered as observable; the second one [DV90a], called in the following T_W , gives a "weak" translation from LTSs to KSs, i.e. a different translation is given if there are transitions labelled by an unobservable action. However, given an LTS \mathcal{A} , all the information that we can find in $T_S(\mathcal{A})$ can be found in $T_W(\mathcal{A})$ and viceversa. Indeed we can easily check that both T_S and T_W preserve the strong bisimulation equivalence, i. e. strongly bisimilar LTSs are mapped into strongly bisimilar KSs and non equivalent LTSs are mapped into non equivalent KSs. The main difference between T_S and T_W relates to the complexity of the translation, in fact T_S requires quadratic time, while T_W requires linear time.

In this context we adopt an extension of the translation T_W , called T'_W [DFGR92], since we are going to reuse the EMC model checker in order to perform model checking on the action-based logics presented above. In fact EMC accepts as input a Kripke Structure with *fair* paths (that is a Kripke Structure in which all fullpaths are infinite in length), hence the original translation T_W has been modified [DFGR92] in order to extend finite fullpaths: if there exist finite fullpaths in the Labelled Transition System, then in the corresponding Kripke Structure a new state q_f is created, with label \perp and successor q_f , and finite fullpaths are extended relating deadlock states with q_f .

4.1 The translation T'_W

In the "weak translation" [DV90a] the construction of the Kripke Structure from a Labelled Transition System is made by splitting transitions labelled by visible actions and creating a new state for each of them, labelled with the label of the original transition; the generated system has almost the same structure of the original one.

Definition 1 (*From LTSs to KSSs*) [DFGR92]

Let $\mathcal{A} = (Q, A_\tau, \rightarrow, 0_Q)$ be a LTS, Q_d the subset of states of Q that have no successors, \perp be a fresh symbol not in A_τ and q_f be a fresh state not in Q . The Kripke Structure, $T'_W(\mathcal{A})$, is defined as $(S', AP, L, \rightarrow', 0_Q)$ where:

- $S' = Q \cup \{(r, a, s) \mid a \in A \text{ and } (r, a, s) \in \rightarrow\} \cup N$, where $N = \{q_f\}$ if $Q_d \neq \{\}$ and $N = \{\}$ otherwise;
- $AP = A \cup \{\perp\}$;
- $\rightarrow' = \{(r, s) \mid (r, \tau, s) \in \rightarrow\} \cup \{(r, (r, a, s)) \mid (r, a, s) \in \rightarrow\} \cup \{((r, a, s), s) \mid (r, a, s) \in \rightarrow\} \cup D$, where $D = \{(q, q_f) \mid q \in Q_d\} \cup \{(q_f, q_f)\}$ if $Q_d \neq \{\}$ and $D = \{\}$ otherwise;
- For $r, s \in Q$ and $a \in A$: $L(r) = L(s) = \{\perp\}$, $L((r, a, s)) = \{a\}$;
- $L(q_f) = \{\perp\}$.

Note that the translation T'_W produces a Kripke Structure with a larger number of states; new states are labelled with the same label of the corresponding transition while old states are labelled with a *fresh* symbol, \perp , whose meaning is: *no visible actions occurred*.

Figure 2 shows how T'_W performs the translation of a given state q of a LTS. Note that the splitting of transitions is restricted to observable transitions.

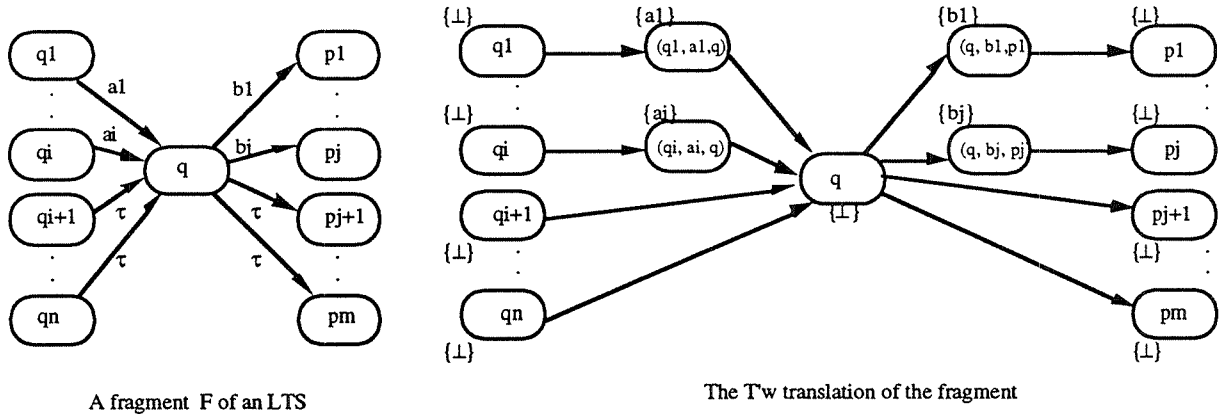


Figure 2

The size of the sets of states and transitions of the Kripke Structure produced by the above algorithm is given by the following formulae, where $n = |Q|$, $d = |Q_d|$, $m = |\rightarrow|$, and u is the number of unobservable transitions in \rightarrow :

$$|S'| = \begin{cases} n + m - u & \text{if } d = 0 \\ n + m - u + 1 & \text{if } d \neq 0 \end{cases} \quad \text{and } |\rightarrow'| = \begin{cases} 2m - u & \text{if } d = 0 \\ 2m - u + 1 + d & \text{if } d \neq 0 \end{cases}$$

5 From Action-based Logics to CTL

The translation T'_W provide a way to relate action-based logics and CTL, characterising CTL as target logic and KSs as underlying models. According to T'_W we define a set of translations, from action-based logics to CTL; we will denote the translation function from \mathcal{J} to CTL with the symbol $\langle \cdot \rangle_{\mathcal{J}}$ (without the index \mathcal{J} if the domain of the function is clear from the context).

5.1 From HML to CTL

The T'_W translation induces a translation of HML formulae into formulae of CTL. A function $\langle \cdot \rangle_H: \text{HML} \rightarrow \text{CTL}$ is defined as follows:

- $\langle \text{true} \rangle = \text{true}$
- $\langle \neg \phi \rangle = \neg \langle \phi \rangle$
- $\langle \phi_1 \wedge \phi_2 \rangle = \langle \phi_1 \rangle \wedge \langle \phi_2 \rangle$
- $\langle \langle a \rangle \phi \rangle = \perp \wedge \text{EX}(a \wedge \text{EX}(\perp \wedge \langle \phi \rangle))$
- $\langle \langle \tau \rangle \phi \rangle = \perp \wedge \text{EX}(\perp \wedge \langle \phi \rangle)$

Proposition 1 Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ be an LTS and $\mathcal{K} = (S, AP, L, \rightarrow', 0_Q)$ be $T'_W(\mathcal{A})$. Then for every $q \in Q$, for every HML formula ϕ , $q \models_{\mathcal{A}} \phi$ if and only if $q \models_{\mathcal{K}} \langle \phi \rangle_H$. (For the proof see the Appendix B)

5.2 From HML' to CTL

Analogously to the previous case, we define a function $\langle \cdot \rangle_{H'}: \text{HML}' \rightarrow \text{CTL}$:

- $\langle \text{true} \rangle = \text{true}$
- $\langle \neg \phi \rangle = \neg \langle \phi \rangle$
- $\langle \phi_1 \wedge \phi_2 \rangle = \langle \phi_1 \rangle \wedge \langle \phi_2 \rangle$
- $\langle \langle \epsilon \rangle \phi \rangle = \text{E}(\perp \cup (\perp \wedge \langle \phi \rangle))$
- $\langle \langle a \rangle \phi \rangle = \perp \wedge \text{EXE}(\perp \cup (a \wedge \text{EXE}(\perp \cup (\perp \wedge \langle \phi \rangle))))$

Proposition 2 Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ be an LTS and $\mathcal{K} = (S, AP, L, \rightarrow', 0_Q)$ be the KS $T'_W(\mathcal{A})$. Then for every $q \in Q$, for every HML' formula ϕ , $q \models_{\mathcal{A}} \phi$ if and only if $q \models_{\mathcal{K}} \langle \phi \rangle_{H'}$. (For the proof see the Appendix B)

5.3 From HML_U to CTL

The translation function $\langle \cdot \rangle_U: \text{HML}_U \rightarrow \text{CTL}$ is so defined:

- $\langle \text{true} \rangle = \text{true}$

- $\langle \neg \phi \rangle = \neg \langle \phi \rangle$
- $\langle \phi_1 \wedge \phi_2 \rangle = \langle \phi_1 \rangle \wedge \langle \phi_2 \rangle$
- $\langle \phi \langle \varepsilon \rangle \phi' \rangle = \mathbf{E} ((\perp \wedge \langle \phi \rangle) \mathbf{U} (\perp \wedge \langle \phi' \rangle))$
- $\langle \phi \langle a \rangle \phi' \rangle = (\perp \wedge \langle \phi \rangle) \wedge \mathbf{E} ((\perp \wedge \langle \phi \rangle) \mathbf{U} (a \wedge \mathbf{EXE}(\perp \wedge \langle \phi' \rangle)))$

Proposition 3 Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ be an LTS and $\mathcal{K} = (S, AP, \mathbf{L}, \rightarrow', 0_Q)$ be the KS $T'_W(\mathcal{A})$. Then for every $q \in Q$, for every HML_U formula ϕ , $q \models_{\mathcal{A}} \phi$ if and only if $q \models_{\mathcal{K}} \langle \phi \rangle_U$. (For the proof see the Appendix B)

5.4 From ACTL to CTL

The mapping $\langle \rangle_A : \text{ACTL} \rightarrow \text{CTL}$ is defined as follows ([DFGR92]):

- $\langle \text{true} \rangle = \text{true}$,
- $\langle \neg \phi \rangle = \neg \langle \phi \rangle$,
- $\langle \phi \wedge \phi' \rangle = \langle \phi \rangle \wedge \langle \phi' \rangle$,
- $\langle \exists (\phi \chi \mathbf{U} \chi' \phi') \rangle = \mathbf{E}(((\perp \wedge \langle \phi \rangle) \vee (\neg \perp \wedge \chi)) \mathbf{U} ((\neg \perp \wedge \chi') \wedge \mathbf{EX}(\perp \wedge \langle \phi' \rangle)))$,
- $\langle \exists (\phi \chi \mathbf{U} \phi') \rangle = (\perp \wedge \langle \phi' \rangle) \vee \mathbf{E}(((\perp \wedge \langle \phi \rangle) \vee (\neg \perp \wedge \chi)) \mathbf{U} (\perp \wedge \langle \phi' \rangle))$,
- $\langle \exists X \chi \phi \rangle = \mathbf{EX}(\neg \perp \wedge \chi \wedge \mathbf{EX}(\langle \phi \rangle))$,
- $\langle \exists X_{\tau} \phi \rangle = \perp \wedge \mathbf{EX}(\perp \wedge \langle \phi \rangle)$,
- $\langle \forall (\phi \chi \mathbf{U} \chi' \phi') \rangle = \mathcal{A}(((\perp \wedge \langle \phi \rangle) \vee (\neg \perp \wedge \chi)) \mathbf{U} ((\neg \perp \wedge \chi') \wedge \mathcal{AX}(\perp \wedge \langle \phi' \rangle)))$,
- $\langle \forall (\phi \chi \mathbf{U} \phi') \rangle = (\perp \wedge \langle \phi' \rangle) \vee \mathcal{A}(((\perp \wedge \langle \phi \rangle) \vee (\neg \perp \wedge \chi)) \mathbf{U} (\perp \wedge \langle \phi' \rangle))$,
- $\langle \forall X \chi \phi \rangle = \mathcal{AX}(\neg \perp \wedge \chi \wedge \mathcal{AX}(\langle \phi \rangle))$,
- $\langle \forall X_{\tau} \phi \rangle = \perp \wedge \mathcal{AX}(\perp \wedge \langle \phi \rangle)$.

Proposition 4 Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ an LTS and $\mathcal{K} = (S, AP, R, 0_Q)$ the KS $T'_W(\mathcal{A})$. Then $0_Q \models_{\mathcal{A}} \phi$ if and only if $0_Q \models_{\mathcal{K}} \langle \phi \rangle_A$. (For the proof see [DFGR92])

This last translation says that CTL is at least as expressive as ACTL. The other logics are respectively in the following relations: HML_U corresponds to ACTL- $\{X\}$ and it is strictly more expressive than HML'; HML, that supports only strong equivalences, is strictly contained into ACTL and it is incomparable w.r.t. HML_U and HML', since they are able to cover weak behaviours.

6 The model checker

The model checker we have realized for the variants of HML, ranging from the original one to ACTL, relies on EMC and permits verifying the validity of a formula in one of such logics on a Labelled Transition System. To perform the check of a formula ϕ of a logic \mathbf{J} on a Labelled Transition System M , the following steps are needed in this framework:

- 1) Translate M into the corresponding Kripke Structure M' , using the translation T'_W ;
- 2) Translate ϕ into the corresponding CTL formula ϕ' , using the proper translation $\langle \rangle_{\mathbf{J}}$;

3) Perform the Model Checking of φ' on M' , using EMC.

Each of these step corresponds therefore to a module of our model checker, called respectively Model Translator, Logic Translator, and EMC itself. The output of the Model Translation phase is input to EMC. Logic expressions to be verified can be given at the EMC prompt; any time a formula is given as input, EMC calls the Logic Translator to get the corresponding CTL formula; then EMC checks the CTL formula on the Kripke Structure, giving as result true or false.

It is not difficult to see that we can perform model checking with time complexity $O((|Q|+|\rightarrow|) \times |\varphi|)$. Indeed, if we let \mathcal{A} be a finite LTS, s be a state of \mathcal{A} and φ be a formula of the logic \mathcal{J} , in order to determine whether $q \models_{\mathcal{A}} \varphi$ it suffices to check whether $q \models_{T_W(\mathcal{A})} \langle \varphi \rangle_{\mathcal{J}}$. We can compute $T_W(\mathcal{A})$ in $O(|Q|+|\rightarrow|)$ -time and the number of states and transitions of $T_W(\mathcal{A})$ will be of order $|Q|+|\rightarrow|$. The formula $\langle \varphi \rangle_{\mathcal{J}}$ can be computed in $O(|\varphi|)$ -time and its size will be of order $|\varphi|$. Model checking of formula φ on \mathcal{A} with the algorithm for CTL of [CES86] can be therefore performed in $O((|Q|+|\rightarrow|) \times |\varphi|)$ -time.

7. Conclusions

We have presented a model checker for the verification of logical properties of systems described by means of process algebras. Logical properties can be expressed in several variants of Hennessy-Milner logics, and they are interpreted over Labelled Transition Systems, which are models also for process algebras. The tool is based on the EMC model checker [CES86] and it employs a set of translation functions, from the variants of HML to the logic CTL, and a translation from Labelled Transition Systems to Kripke Structures.

We consider that the approach of reusing existing tools has proven effective for obtaining a rapid prototype environment. In fact, EMC was chosen since we considered it as the only established and widely used model checker. In general, we aimed at relying on widely used and continuously upgraded tools, for which new research is undertaken to deal with unsolved problem, like that of state explosion and symbolic model checking [BCDMH90].

A similar approach of reusing existing model checkers is presented in [JKP90]. There CTL is used as a logic for Labelled Transition Systems by modifying the satisfaction relation; a relativized satisfaction relation $\langle a, s \rangle \models \varphi$ is introduced. The satisfaction relation we have given for the action based logics are more immediate. Besides, in [JKP90] invisible actions are not considered. Also a different translation from Labelled Transition Systems to Kripke Structures is used that is not linear with the number of transitions of the original transition system.

Alternatively, model checking for action based-logics could be performed using a μ -calculus model checker; algorithms for performing linear model checking for fragments of the μ -calculus corresponding to CTL have been presented [CS91] and their implementation is planned in the Concurrency Workbench [CPS90].

Extensions of our environment are currently under study, and integration with other tools will be considered. The first extension we will provide is the "counterexample" facility provided by EMC for CTL formula. EMC, whenever a formula φ does not hold, produces a path in the model which falsifies φ and generally provides useful information on how to modify the model to have the formula satisfied. The direct correspondence between the Labelled Transition System and the generated

Kripke Structure, should allow us to easily determine the failing path in a Labelled Transition System by analyzing the path provided by EMC for the generated Kripke Structure.

Acknowledgements

We would like to thank Rocco De Nicola for joint work and discussions on the topics of the paper.

REFERENCES

- [BCG88] M. C. Browne, E. M. Clarke, O. Grumberg, "Characterizing Finite Kripke Structures in Propositional Temporal Logic", *Theoretical Computer Science*, vol. 59, pp. 115-131, 1988.
- [BCMDH90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, J. Hwang: Symbolic Model Checking: 10^{20} states and beyond. Proc. of the 5th Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, USA, June 1990, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 428-439.
- [BGS88] A. Boujani, S. Graf, J. Sifakis: A Logic for the Description of Behaviours and Properties of Concurrent Systems. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, (de Bakker, J., de Roever, P. and Rozenberg, G. eds.) *Lecture Notes in Computer Science 354*, Springer-Verlag, 1989, pp. 398-410.
- [CES86] E.M. Clarke, E.A. Emerson, A.P. Sistla: "Automatic Verification of Finite State Concurrent Systems using Temporal Logic Specifications", *ACM TOPLAS* 8, 2 April 1986.
- [CPS90] R. Cleaveland, J. Parrow, B. Steffen: "The Concurrency Workbench". In: *Automatic Verification Methods for Finite State Systems* (J. Sifakis, ed.) *LNCS 407*, Springer-Verlag, 1990, pp. 24-37.
- [CS91] R. Cleaveland, B. Steffen: "A Linear-Time Model Checking Algorithm for the Alternation-Free Modal μ -Calculus", *Proc. of 3rd Workshop on Computer Aided Verification*, July 1-4, 1991, Aalborg, Denmark.
- [DFGR92] R. De Nicola, A. Fantechi, S. Gnesi, G. Ristori: "An action-based framework for verifying logical and behavioural properties of concurrent systems". *Proc. of 3rd Workshop on Computer Aided Verification*, July 1-4, 1991, Aalborg, Denmark, *LNCS 575*; to appear also in *Computer Networks and ISDN Systems*, North-Holland, 1992.
- [DV90a] R. De Nicola, F. Vaandrager: "Action versus State based Logics for Transition Systems", in *Proceedings Ecole de Printemps on Semantics of Concurrency* (I. Guessarian ed.), April 1990, *LNCS 469*, Springer Verlag .
- [DV90b] R. De Nicola, F. Vaandrager: "Three Logics for Branching Bisimulations" in *Proc. of the 5th Annual Symposium on Logic in Computer Science (LICS '90)*, Philadelphia, USA, June 1990, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 118-129.
- [EH86] E. A. Emerson, J. Halpern, ""Sometime" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic", *Journal of ACM*, vol. 33, January 1986, pp. 151-178.
- [ES89] E. A. Emerson, J. Srinivasan: "Branching Time Temporal Logic". In: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, (de Bakker et al., eds.) *LNCS 354*, Springer-Verlag, 1989, pp. 123-172.
- [HM85] M. Hennessy, R. Milner, "Algebraic Laws for Nondeterminism and Concurrency", *Journal of*

ACM, vol. 32, n. 1, January 1985, pp. 137-161.

- [JKP90] B. Jonsson, A. H. Khan, J. Parrow, "Implementing a Model Checking Algorithm by Adapting Existing Automated Tools", In: Automatic Verification Methods for Finite State Systems (J. Sifakis, ed.) LNCS 407, Springer-Verlag, 1990, pp. 179-188.
- [Lar90] K. G. Larsen: Proof Systems for Satisfiability in Hennessy-Milner Logic with Recursion, Theoretical Computer Science, 72 (2),1990, pp.265-288.
- [Mil80] R. Milner, "A Calculus of Communicating Systems", LNCS 92, Springer-Verlag, 1980.
- [Mil83] R. Milner, "Calculi for Synchrony and Asynchrony", Theoretical Computer Science 25, 1983.
- [MP89] Z. Manna, A. Pnueli, "The Anchored Version of the Temporal Framework", in: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, (de Bakker et al., eds.) LNCS 354, Springer-Verlag, 1989.
- [Par81] D. Park "Concurrency and automata on infinite sequences", Proceedings 5th GI Conf., LNCS 104, Springer-Verlag, 1981, pp. 167-183.
- [Pnu85] A. Pnueli, "Linear and Branching Structures in the Semantics and Logic of Reactive Systems" Proceedings of 12th ICALP, LNCS 194, Springer-Verlag, 1985.
- [Sti90] C. Stirling, "An Introduction to Modal and Temporal Logics for CCS", in: Concurrency: Theory, Language, and Architecture, (A. Yonezawa, T. Ito, eds.) LNCS 491, Springer-Verlag, 1990, pp. 2-20.
- [vGW89] R. J. van Glabbeek, W. P. Weijland: "Branching Time and Abstraction in Bisimulation Semantics". In: Information Processing '89 (G.X. Ritter, ed.), North Holland, 1989, pp. 613-618.

Appendix A

A binary relation R on an LTS \mathcal{A} is called a *(strong) bisimulation* if it is symmetric and satisfies the following:

if rRs and $r \xrightarrow{\alpha} r'$, then $\exists s' : s \xrightarrow{\alpha} s'$ and $r'Rs'$

Two states p and q of \mathcal{A} are said *(strongly) bisimulation equivalent* (written $p \approx q$) if and only if there exists a bisimulation R with pRq [Par 81, Mil 83].

A binary relation R on \mathcal{A} is called a *weak bisimulation* if it is symmetric and satisfies the following:

if rRs and $r =\alpha=> r'$, then $\exists s' : s =\alpha=> s'$ and $r'Rs'$.

Two states p and q of a \mathcal{A} are said *weakly bisimulation equivalent* (written $p \sim q$), or *observational equivalent*, if and only if there exists a weak bisimulation R with pRq [Mil80].

A binary relation R on \mathcal{A} is called a *branching bisimulation* if it is symmetric and satisfies the following:

if rRs and $r \xrightarrow{\alpha} r'$, then either $\alpha=\tau$ and $r'Rs$, or $\exists s_1, s' : s =\epsilon=> s_1 \xrightarrow{\alpha} s'$ and $rRs_1, r'Rs'$.

Two states p and q of \mathcal{A} are said *branching bisimulation equivalent* (written $p \approx_b q$), if and only if there exists a branching bisimulation R with pRq [vGW 89].

Appendix B

All the proofs of Propositions 1, 2 and 3 are given by applying structural induction on formulae and by taking into consideration the translation T'_W from LTSs to KSs proposed in Section 4.

We stress here some facts about the translation T'_W . Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ be the LTS to be translated and $\mathcal{K} = (S', AP, L, \rightarrow', 0_Q)$ be $T'_W(\mathcal{A})$. First, every state in Q has a correspondent state in S with the same name; moreover, in \mathcal{K} only states corresponding to states in Q are labelled with \perp . New states in \mathcal{K} are created in correspondence of every observable transition and are labelled with the action labelling the transition. It is worth noting that if the label (an action) of a transition satisfies a given action formula χ then the correspondent state in \mathcal{K} satisfies the proposition χ .

Since the logics that we consider allow the negation of formulae, to prove: $q \models_{\mathcal{A}} \phi$ if and only if $q \models_{\mathcal{K}} (\phi)$, we need to show that:

- 1) $q \models_{\mathcal{A}} \neg\phi$ if and only if $q \models_{\mathcal{K}} (\neg\phi)$ and
- 2) $q \models_{\mathcal{A}} \phi$ implies $q \models_{\mathcal{K}} (\phi)$.

Here only the proof of Proposition 1 is shown, in order to give a flavour of the way we apply induction in the proofs of Propositions 1, 2 and 3. The other proofs are boring applications of the same principle used in Proposition 1.

Proposition 1: Let $\mathcal{A} = (Q, A, \rightarrow, 0_Q)$ be an LTS and $\mathcal{K} = (S, AP, \mathbb{L}, \rightarrow', 0_Q)$ be $T'_{\mathbb{W}}(\mathcal{A})$. Then for every $q \in Q$, for every HML formula ϕ , $q \vDash_{\mathcal{A}} \phi$ if and only if $q \vDash_{\mathcal{K}} \langle \phi \rangle_H$.

Proof: The proof is based on structural induction on HML formulae.

case $\phi = \mathbf{true}$

$\langle \mathbf{true} \rangle = \mathbf{true}$. We have always $q \vDash_{\mathcal{A}} \mathbf{true}$ and $q \vDash_{\mathcal{K}} \mathbf{true}$.

case $\phi = \neg \phi'$

$\langle \neg \phi' \rangle = \neg \langle \phi' \rangle$. We have that $q \vDash_{\mathcal{A}} \neg \phi'$ is true if and only if $q \not\vDash_{\mathcal{A}} \phi'$ holds and this is true if and only if, applying the inductive hypothesis, $q \not\vDash_{\mathcal{K}} \langle \phi' \rangle$, that is if and only if $q \vDash_{\mathcal{K}} \neg \langle \phi' \rangle$.

case $\phi = \phi_1 \wedge \phi_2$

$\langle \phi_1 \wedge \phi_2 \rangle = \langle \phi_1 \rangle \wedge \langle \phi_2 \rangle$. If $q \vDash_{\mathcal{A}} \phi_1 \wedge \phi_2$ then $q \vDash_{\mathcal{A}} \phi_1$ and $q \vDash_{\mathcal{A}} \phi_2$. Applying the inductive hypothesis, this implies that $q \vDash_{\mathcal{K}} \langle \phi_1 \rangle$ and $q \vDash_{\mathcal{K}} \langle \phi_2 \rangle$, that is $q \vDash_{\mathcal{K}} \langle \phi_1 \rangle \wedge \langle \phi_2 \rangle$.

case $\phi = \langle a \rangle \phi'$

$\langle \langle a \rangle \phi' \rangle = \perp \wedge \mathbf{EX}(a \wedge \mathbf{EX}(\perp \wedge \langle \phi' \rangle))$.

If $q \vDash_{\mathcal{A}} \langle a \rangle \phi'$ then there exists $q' \in Q$ such that $q \xrightarrow{a} q'$ and $q' \vDash_{\mathcal{A}} \phi'$; it follows that a new state (q, a, q') is created in \mathcal{K} and the following relations are established:

$(q, (q, a, q')) \in \rightarrow'$, $((q, a, q'), q') \in \rightarrow'$. Also, $\perp \in \mathbb{L}(q)$, $\perp \in \mathbb{L}(q')$ and $a \in \mathbb{L}((q, a, q'))$.

Applying the inductive hypothesis, $q' \vDash_{\mathcal{A}} \phi'$ implies $q' \vDash_{\mathcal{K}} \langle \phi' \rangle$. Hence, $q \vDash_{\mathcal{K}} \perp \wedge \mathbf{EX}(a \wedge \mathbf{EX}(\perp \wedge \langle \phi' \rangle))$.

case $\phi = \langle \tau \rangle \phi'$

$\langle \langle \tau \rangle \phi' \rangle = \perp \wedge \mathbf{EX}(\perp \wedge \langle \phi' \rangle)$

$q \vDash_{\mathcal{A}} \langle \tau \rangle \phi'$ means that there exists a state $q' \in Q$ such that $q \xrightarrow{\tau} q'$ and $q' \vDash_{\mathcal{A}} \phi'$. This implies that in \mathcal{K} $(q, q') \in \rightarrow'$ and $\perp \in \mathbb{L}(q)$, $\perp \in \mathbb{L}(q')$. For the inductive hypothesis $q' \vDash_{\mathcal{K}} \langle \phi' \rangle$, hence $q \vDash_{\mathcal{K}} \perp \wedge \mathbf{EX}(\perp \wedge \langle \phi' \rangle)$. \square