Temporal Deductive Verification of Cache Coherence Protocols*

Michael Fisher and Alexei Lisitsa
Department of Computer Science
University of Liverpool, Liverpool, United Kingdom
Email: {M.Fisher,A.Lisitsa}@csc.liv.ac.uk

Abstract

First-order temporal logics (FOTLs) are very expressive formalisms in which we can specify, at a very natural level of abstraction, (almost) any computational system, together with its correctness conditions. This makes FOTLs ideal for formally specifying both software and hardware systems. Unfortunately, the expressiveness of FOTL also means that such logics are typically highly undecidable. Consequently, the fully automated deductive verification of FOTL specifications is not, in general, possible.

Recently, however, a tractable fragment of FOTL was identified by Hodkinson, Wolter and Zakharyashev. This fragement, termed the *monodic* fragment is amenable to automated proof techniques, and we have developed a number of theorem-proving systems for this fragment. In this paper, we particularly consider the use of such tractable fragments of first-order temporal logics in the deductive verification of an important class of protocols, namely *cache coherence* protocols. This is the first use of monodic FOTLs in the deductive verification of reactive systems.

1 Introduction

Local cache techniques [10] are important in shared-memory multiprocessor systems because they allow the reduction of both memory access latency and network traffic. Using such techniques, each processor has its own fast memory cache, holding local copies of main memory blocks. Although reducing the access time, this approach poses the problem of cache consistency, whereby one has to ensure that copies of the *same* memory block residing in the caches of *different* processors are consistent. Such data consistency can be provided by *cache coherence*

*This work was supported by EPSRC, through research grant GR/M46631.

protocols, which typically operate as follows: every processor is equipped with a finite state control process, which *reacts* to *read* and *write* requests. This control process ensures that consistency of each cache is retained with respect to the other caches in the system. Abstracting from the low-level implementation details of the I/O and synchronization primitives, one may consider cache coherence protocols as families of identical finite state machines together with a primitive form of communication. Correctness at such a level of abstraction can be expressed in terms of co-occurrences of some states in which, for example, the processors are allowed to change the content of their cache copies.

While cache coherence protocls have many practical advantages, their use is dependent on the fact that they correctly enforce coherence. In some cases, especially when a large number of caches are considered, this correctnes may not be immediately obvious. Hence, there has been a more to use formal verification techniques in order to ascertain the protocol's correctness. The paper [15] provides a good survey of verification methods that have been used to prove correctness of cache coherence protocols. If the number of processors is fixed, verification can be carried out using traditional model checking methods [2]. More challenging is the problem of the verification of cache coherence protocols for *every* possible number of processors (termed "parameterized cache protocols"). In [5, 6] *parameterized model checking*, based on constraint technology, has been applied and has been shown to be a method that is complete for a certain class of protocols and correctness conditions.

In this paper we adopt an alternative approach. We show how one may verify cache protocols for all possible dimensions by first translating their specifications into *monodic first-order temporal specifications* [3, 1] and then applying temporal theorem proving techniques [8, 7, 4]. In order to achieve this, we provide a faithful translation from the finite state model of cache protocols [14] to monodic temporal specifications [3, 1] and demonstrate two proof methods suitable for the proof of correctness of cache coherence protocols. The applicability of these methods is illustrated by the proof of correctness of two different cache coherence protocols, namely the "Synapse N+1" and "Illinois" protocols [5].

2 Modelling Cache Protocols

We here consider how cache coherence protocols are modelled formally, and how verification can be carried out on such models.

2.1 Finite State Machine Model

In [5], the following finite state machine based model, suitable for the specification and verification of cache coherence protocols, was defined. Essentially, it uses a family of identical finite state automata together with a rudimentary form of communication: if one automaton makes a transition (an action) a, then it is requried that *all* other automata simultaneously make a complementary transition (reaction) $\bar{a}$. (This can be seen as a form of synchronous broadcast communication.)

Definition 1 (Protocol) *A protocol, $\mathcal{P}$ is a tuple $\langle Q, \Sigma, \bar{\Sigma}, \tau \rangle$, where*

- *Q is a finite set of states;*
- *Σ is a finite set of actions;*
- *$\bar{\Sigma} = \{\bar{\sigma} \mid \sigma \in \Sigma\}$ is the set of reactions;*
- *$\tau : Q \times \{\Sigma \cup \bar{\Sigma}\} \times Q$ is a transition relation that satisfies the following property*
$$\forall \sigma \in \Sigma.\ \forall q \in Q.\ \exists q' \in Q.\ \langle q, \bar{\sigma}, q' \rangle \in \tau$$

Definition 2 (Global Machine) *Given a protocol, $\mathcal{P} = \langle Q, \Sigma, \bar{\Sigma}, \tau \rangle$, the global machine $\mathcal{M}_G$ of dimension n is the tuple $\langle Q_{\mathcal{M}_G}, \Sigma_{\mathcal{M}_G}, \tau_{\mathcal{M}_G} \rangle$, where*

- $Q_{\mathcal{M}_G} = Q^n$
- $\Sigma_{\mathcal{M}_G} = \Sigma$
- *$\tau_{\mathcal{M}_G} : Q_{\mathcal{M}_G} \times \Sigma_{\mathcal{M}_G} \times Q_{\mathcal{M}_G}$ is a transition relation that satisfies the following property*
$\langle \langle s_1, \ldots, s_n \rangle, \sigma, \langle s'_1, \ldots, s'_n \rangle \rangle \in \tau_{\mathcal{M}_G}$ *iff* $\exists i.\ \langle s_i, \sigma, s'_i \rangle \in \tau$ *and* $\forall j \neq i.\ \langle s_j, \bar{\sigma}, s'_j \rangle \in \tau$

An element $G \in Q_{\mathcal{M}_G}$ is said to be a global state. A run of a global machine $\mathcal{M}_G$ is a possibly infinite sequence $G_1, G_2, \ldots G_i \ldots$ of global states of $\mathcal{M}_G$ such that for all i, $\langle G_i, G_{i+1} \rangle \in \tau_{\mathcal{M}_G}$.

2.2 Finite State Machine Model with global conditions

Not all cache coherence protocols can be specified in the above model. An useful extension is a model with additional *global conditions* [5], where conditional actions of the form $P \to \sigma$ can be specified. In that case a transition $\langle S, \sigma, S' \rangle$ is allowed only if the global state S satisfies P. More precisely we have the following definition.

Definition 3 (Global conditions) *Given a set Q of states, a* global condition *over Q defined as follows:*

- $\#q = c$, $\#q \geq c$, $true$ *where $q \in Q$ and $n \in N$ are global conditions.*
- *if ϕ and ψ are global conditions then $\phi \wedge \psi$ and $\phi \vee \psi$ are global conditions.*

For a given Q denote by GC_Q a set of all global conditions for Q.

Definition 4 (Satisfiability of predicates in global states) *For a global state $\bar{q} = \{q_1, \ldots, q_k\} \in Q^k$ and a set of global conditions, inductively define the satisfiability relation, '$\models$', as follows*[1]*:*

- $\bar{q} \models \#q = c \quad$ *iff* $\quad Card\{i \mid q_i = q\} = c$;
- $\bar{q} \models \#q \geq c \quad$ *iff* $\quad Card\{i \mid q_i = q\} \geq c$;
- $\bar{q} \models \phi \wedge \psi \quad$ *iff* $\quad \bar{q} \models \phi$ *and* $\bar{q} \models \psi$;
- $\bar{q} \models \phi \vee \psi \quad$ *iff* $\quad \bar{q} \models \phi$ *or* $\bar{q} \models \psi$

Definition 5 (Protocol with global conditions) *A protocol $\mathcal{P}$ with global conditions is a tuple $\langle Q, \Sigma, \bar{\Sigma}, C, \tau \rangle$, where*

- *Q is a finite set of states;*
- *Σ is a finite set of actions;*
- *$\bar{\Sigma} = \{\bar{\sigma} \mid \sigma \in \Sigma\}$ is the set of reactions;*
- *$C \subseteq GC_Q$ is a finite set of global conditions over Q*
- *$\tau : Q \times \{(C \times \Sigma) \cup \bar{\Sigma}\} \times Q$ is a transition relation that satisfies the following property*

$$\forall (P \to \sigma) \in C \times \Sigma.\, \forall q \in Q.\, \exists! q' \in Q.\, \langle q, \bar{\sigma}, q' \rangle \in \tau$$

Definition 6 (Global machine with global conditions) *Given a protocol with global conditions $P = \langle Q, \Sigma, \bar{\Sigma}, C, \tau \rangle$, the global machine $\mathcal{M}_G$ of dimension n is the tuple*
$\langle Q_{\mathcal{M}_G}, \Sigma_{\mathcal{M}_G}, \tau_{\mathcal{M}_G} \rangle$, where

- $Q_{\mathcal{M}_G} = Q^n$

[1] '$Card$' calculates the cardinality of a set.

- $\Sigma_{\mathcal{M}_G} = \Sigma$
- $\tau_{\mathcal{M}_G} : Q_{\mathcal{M}_G} \times \Sigma_{\mathcal{M}_G} \times Q_{\mathcal{M}_G}$ *is a transition relation satisfying the property that*

  $\langle\langle s_1, \ldots, s_n\rangle, \sigma, \langle s'_1, \ldots, s'_n\rangle\rangle \in \tau_{\mathcal{M}_G}$ *iff* $\exists i.\ \langle s_i, P \to \sigma, s'_i\rangle \in \tau$, $\langle s_1, \ldots, s_n\rangle \models P$, *and* $\forall j \neq i. \langle s_j, \bar{\sigma}, s'_j\rangle \in \tau$

A global state and a run of the machine is defined in the same way as in Definition 2 above.

Note 1. It follows from the above definitions that the simple model from Definition 1 is a particular case of the more general model with global conditions, in which *all* conditions are given by the formula *true*.

2.3 Verification

Verification of a cache coherence protocol specified as a global machine of the above form amounts to showing that any run which starts in a global state, satisfying some initial condition ψ, satisfies some *correctness* property χ.

If the dimension of the machine is fixed, say $n = 5$, then verification can be carried out using traditional model checking approaches [2]. However, more advanced methods are required if we wish to verify correctness of a protocol for *every* possible dimension (analogously, for any size of collection of finite state processors). In [5, 6], a *parametrized model checking* technique based on constraint technology has been applied to this problem, and it has been shown that the method is complete for particular classes of protocols and correctness conditions. However, we here use a deductive approach, providing a first-order temporal description of the protocols and verifying their correctness conditions using monodic temporal theorem-proving.

The two following kinds of correctness conditions are typical, and usually vital, for cache coherence protocols.

Non co-occurence of states: some local states $q_1, q_2 \in Q$ can not appear in the same global state (i.e. separate processors cannot simultaneously be in states q_1 and q_2).

At most one: a local state q can appear at most once (i.e. be a state of at most one processor) in any global state.

3 Monodic Temporal Specifications

We assume that the language $\mathcal{TL}$ of the first-order temporal logic is constructed in the standard way (see i.e. [9, 11]) from a classical (non-temporal) first-order language $\mathcal{L}$ and a set of future-time temporal operators '$\Diamond$' (*sometime in the future*), '$\Box$' (*always in the future*), and '$\bigcirc$' (*in the next moment*). Here, $\mathcal{L}$ does not contain functional symbols. Formulae of $\mathcal{TL}$ are interpreted in *first-order temporal structures* with constant domains. We assume a *rigid* semantics for variables and constants, i.e. the intepretation of these elements does not change over time. The standard notions of *satisfiable* and *valid* formulae (evaluated in initial worlds of temporal structures) are used.

A first-order temporal specification is a triple $< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ where $\mathcal{U}$ and $\mathcal{S}$ are termed the *universal part* and the *initial part*, respectively. These two parts are given as finite sets (conjunctions) of (nontemporal) first-order formulae. In contrast, the *temporal part*, $\mathcal{T}$, is given by a finite set (conjunction) of *temporal step formulae*. A temporal step formula has the following form:

$$\forall \bar{x}\ (\varphi(\bar{x}) \rightarrow \bigcirc\chi(\bar{x}))$$

where φ and χ are first-order formulae with the (possibly empty) sequence of free variables $\bar{x}$.

Note 2. A temporal specification $< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ is just an alternative form of representation for the temporal formula $\mathcal{S} \wedge \Box(\mathcal{U} \wedge \mathcal{T})$.

If every temporal step formula in the specification SP is in the form $\forall x\ (P(x) \Rightarrow \bigcirc R(x))$ or $p \Rightarrow \bigcirc r$, where P and p are unary (i.e. one-place) predicate and propositional symbols, respectively, while $R(x)$ and r are boolean expressions composed from one-place predicate and propositional symbols, respectively, then the specification is termed *monodic* [11, 1].

Definition 7 (Ground Induction Problem) *Given a first-order temporal specification SP and a ground first-order formula ψ,* a ground induction problem *is one where the validity of the formula $SP \rightarrow \Box\psi$ must be established.*

4 Temporal Specifications for Parametrized Finite State Models

4.1 Temporal Translation

Given a protocol $\mathcal{P} = \langle Q, \Sigma, \bar{\Sigma}, C, \tau\rangle$, we define its translation to a temporal specification (not yet monodic) as follows.

For each $q \in Q$, introduce a monadic predicate symbol P_q and for each $\sigma \in \Sigma$ introduce a monadic predicate symbol A_σ. Introduce one more monadic predicate symbol, A (representing the notion of the "active" automaton). Intuitively, elements of the domain in the temporal representation will represent exemplars of finite automata and the formula $P_q(x)$ is intended to represent "automaton x is in state q".

We further define $SP_{\mathcal{P}} =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$, where[2]

- $\mathcal{U}$ includes the formulae:
  - $\exists x.\, A(x)$ (*there exists an active automaton*)
  - $\forall x.\, \forall y.\, (A(x) \wedge A(y)) \rightarrow (x = y)$ (*at most one automaton is active*)
  - $\forall x.\, A(x) \wedge P_q(x) \leftrightarrow A_{\sigma_{i_1}} \mid A_{\sigma_{i_2}} \mid \ldots \mid A_{\sigma_{i_k}}$ where $\{\sigma_{i_1}, \ldots \sigma_{i_k}\} = \{\sigma \in \Sigma \mid \exists q' (q, \sigma, q') \in \tau\}$ (*active automaton in a state performs one of the actions available*)
  - $\forall x.\, P_{q_1} \mid P_{q_2} \ldots \mid P_{q_m}$ for $Q = \{q_1, \ldots, q_m\}$ (*every automaton is in one of the prescribed states*)

- $\mathcal{S} = \emptyset$ (*no assumptions on the initial global state in the general model; for the purpose of verification $\mathcal{S}$ will be extended by initial conditions*)

- $\mathcal{T}$ includes the formulae:
  - $\forall x.\, P_q(x) \wedge A_\sigma(x) \wedge \Psi_P \rightarrow \bigcirc P_{q'}(x)$ for all q, σ, q' such that $(q, P \rightarrow \sigma, q') \in \tau$ (*the state is changed after an action*)
  - $\forall x.\, P_q(x) \wedge \neg A_\sigma(x) \wedge \exists y A_\sigma(y) \rightarrow \bigcirc P_{q'}(x)$ for all q, σ, q' such that $q' = \tau(q, \bar{\sigma})$ (*the state is changed after a re-action*)
  - $\forall x.\, P_q(x) \wedge \neg \exists y A(y) \rightarrow \bigcirc P_q(x)$ for all $q \in Q$ (*the state is not changed if there is no active automaton*)

Here Ψ_P stands for the first-order formula (with equality) expressing the condition P on a global state:

[2]Note that '|' here stands for "exclusive OR".

- $\Psi_{true} \equiv true$
- $\Psi_{\#q \geq c} \equiv \exists x_1 \ldots \exists x_c (\wedge_{1 \leq i < j \leq c} x_i \neq x_j)$
- $\Psi_{\#q = c} \equiv \Psi_{\#q \geq c} \wedge \forall x_1 \ldots \forall x_c \forall x_{c+1} (\vee_{1 \leq i < \leq j \leq c+1} x_i = x_j)$
- $\Psi_{P1 \wedge P2} \equiv \Psi_{P1} \wedge \Psi_{P2}$
- $\Psi_{P1 \vee P2} \equiv \Psi_{P1} \vee \Psi_{P2}$

This translation of protocol $\mathcal{P}$ is faithful in the following sense.

Proposition 1 *Given a protocol, $\mathcal{P}$, and a global machine, $\mathcal{M}_G$, of dimension n, then any temporal model $M_1, M_2, \ldots$ of $SP_{\mathcal{P}}$ with the finite domain $c_1, \ldots c_n$ of size n represents some run $G_1, G_2, \ldots$ of $\mathcal{M}_G$ as follows:*

$$\text{for all } j.\ \langle q_1, \ldots q_n \rangle \in G_j \quad \leftrightarrow \quad M_j \models P_{q_1}(c_1) \wedge \ldots P_{q_n}(c_n)$$

Dually, for any run of $\mathcal{M}_G$ there is a temporal model of $SP_{\mathcal{P}}$ with a domain of size n representing this run.

Proof By routine inspection of the definitions of runs, temporal models and the translation. ❑

Note 3. If we have a protocol specified as a finite-state automaton with global conditions, then $\mathcal{T}$ will include the formulae $\forall x.\ P_q(x) \wedge A_\sigma(x) \wedge \phi_P \rightarrow \bigcirc P_{q'}(x)$, where ϕ_P is a first-order (monadic) formula expressing condition P for an action σ.

4.2 Monodic specification

The above temporal translation leads to a specification which is not, in general, monodic (as we defined it earlier). However, this specification represents a monodic temporal formula [11], i.e a formula with at most one free variable in any subformula whose main connective is temporal. In this case, we can translate any specification into a monodic one using renaming [16, 9]. Thus, any occurrence of a subformula of the form $P_q(x) \wedge \neg A_\sigma(x) \wedge \exists y.\ A_\sigma(y)$ is replaced by a new predicate symbol $P_{q,\sigma}(x)$ and the universal part of the specification is extended by $\forall x.\ P_{q,\sigma}(x) \leftrightarrow P_q(x) \wedge \neg A_\sigma(x) \wedge \exists y.\ A_\sigma(y)$.

The resulting specification is monodic and remains equivalent to the original one in the sense that it has the same temporal models if we omit interpretations of the newly introduced predicate symbols.

4.3 Correctness and Verification

Temporal representations of correctness conditions described above can be given as:

Non-co-occurence (i.e. no two automata are simultaneously in q_1 and q_2)

$$\Box\ \forall x, y.\ \neg P_{q_1}(x) \lor \neg P_{q_2}(y)$$

At-most-one (i.e. at most one automaton is in state q)

$$\Box\ \forall x, y.\ (P_q(x) \land P_q(y)) \rightarrow (x = y)$$

Many protocols also require some intitial conditions, e.g. all automata are in the same state q, which can be represented by $\forall x.\ P_q(x)$. In general, we allow arbitrary first-order monadic formulae (with equality) in the specification's initial conditions.

In order to verify a cache coherence protocol, $\mathcal{P}$, for an arbitrary number of processors (i.e. a global machine of any dimension), specified by a finite state model, one can proceed as follows.

- translate $\mathcal{P}$ into temporal specification $SP_{\mathcal{P}} =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$;
- add initial conditions (if any) to $\mathcal{S}$;
- for the temporal representation, χ, of a required correctness condition, check whether $SP_{\mathcal{P}} \rightarrow \chi$ is a valid formula.

Thus, verification is reduced to a ground induction problem (as we defined it earlier), which can now be tackled by theorem proving.

In [3, 4], the theorem proving methods for monodic specifications based on clausal temporal resolution [7] have been developed. These methods mainly deal with specifications *without* equality. In [1] a complete method for ground induction problems without equality has been demonstrated.

In both our translation of the protocol and the correctness conditions required above we make use of equality, although in very restricted way. While we do not yet have a completeness result for such specifications in general, it turns out that we can use methods of [1] to prove **non-co-occurence** conditions for cache coherence protocols but can not directly use these methods for **at-most-one** conditions. One of the possible explanations is that these methods are sensitive to the equality used in the conclusion of the ground induction problem, but less sensitive to the use of equality in universal and initial parts of specifications.

5 Proof Methods

5.1 Complete Method for the Case Without Equality

In [1], a complete proof method for the ground induction problem for monodic specifications has been developed and applied. It is derived from the clausal resolution method for propositional and first-order temporal logics [8, 7, 3, 4]. For simplicity we describe the case without constants and propositional variables.

Given a monodic specification $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} = \{P_1 \Rightarrow \bigcirc R_1, \dots P_k \Rightarrow R_k\} >$, a *simple merged rule* for SP is a rule in one of the following forms

- $\Box\forall x(P_{i_1}(x) \vee \dots \vee P_{i_l}(x)) \rightarrow \bigcirc\forall x(R_{i_1} \vee \dots \vee R_{i_l}(x))$ for some $\{i_1, \dots i_l\} \subseteq \{1, \dots, k\}$
- $\Box\exists x(P_{i_1}(x) \wedge \dots \wedge P_{i_l})(x)) \rightarrow \bigcirc\exists x(R_{i_1} \wedge \dots \wedge R_{i_l}(x))$ for some $\{i_1, \dots i_l\} \subseteq \{1, \dots, k\}$

A *merged rule* is then a rule of the form $A \rightarrow \bigcirc B$ which is (equivalent to) a boolean combination of simple merged rules. A specification $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ is said to be reduced if, for any merged rule $A \rightarrow \bigcirc B$ the following condition is satisfied:

$$\text{if } \mathcal{U} \wedge B \vdash \perp \text{ then } \mathcal{U} \wedge A \vdash \perp.$$

Any monodic temporal specification SP can be effectively transformed into an equivalent reduced specification [3, 1].

Theorem 1 (Completeness of the invariant scheme [1]) *Let $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ be a reduced monodic temporal specfication and ψ be a ground first-order formula. Then $\Box\psi$ is a (temporal) logical consequence of SP, i.e. $SP \models \Box\psi$ if, and only if, there exists a set of merged rules $\{A_1 \rightarrow \bigcirc B_1, \dots, A_m \rightarrow \bigcirc B_m\}$ based on $\mathcal{T}$ such that*

- $\mathcal{U} \wedge B_i \vdash \psi$ *for* $i = 1 \dots m$
- $\mathcal{U} \wedge B_i \vdash \bigvee_{i=1}^{m} A_i$ *for* $i = 1 \dots m$
- $\mathcal{S} \wedge \mathcal{U} \vdash \psi \wedge \bigvee_{i=1}^{m} A_i$

In other words the formula $\bigvee_{i=1}^{m} A_i$ from the above theorem is an invariant which is sufficient to prove $\Box\psi$.

We will exhibit the applicability of this method on the verification of a parametrized cache coherence protocols in Section 6, where a **non-co-occurence** property is proved for the "Synapse N+1" and "Illinois" protocols [5]. As mentioned above, it turns out that this method as it is can not be used for proving **at-most-one** properties. However, one may use the invariants produced to establish the **at-most-one** property by simple temporal induction.

5.2 Simple Scheme with Invariants

Given a monodic specification $SP = \langle \mathcal{U}, \mathcal{S}, \mathcal{T} \rangle$, let I be some first-order non-temporal formula invariant for SP, that is $SP \models \Box I$. Let ϕ be a first-order (non-temporal) formula. Then we have the following temporal induction scheme:

Proposition 2 *If* $\mathcal{U} \wedge \mathcal{T} \wedge I \wedge \phi \vdash_{tl} \bigcirc\phi$ *then* $SP \models \Box\phi$.

Proof By obvious induction on the flow of time. ❑

This induction scheme is not guaranteed to be complete, but has proved to be useful for proving **at-most-one** properties if one takes as I an invariant from the scheme of Theorem 1. Notice that the scheme assumes derivability in temporal logic. Establishing that condition can be reduced, however, to just first-order non-temporal derivability by introducing an explicit temporal parameter into formulae, such that:

- the arities of all participating predicates are extended by 1;
- any formula ψ from $\mathcal{U}$ is replaced by $\forall t\psi(t)$, where t is a newly introduced parameter; denote the resulting set of formulae by $\mathcal{U}_t$;
- any formula $\forall x\ (P(x) \Rightarrow \bigcirc R(x))$ from $\mathcal{T}$ is replaced by $\forall t\forall x\ (P(x,t) \Rightarrow R(x, s(t)))$, where s is a newly introduced unary functional symbol; denote the resulting set of step rules by $\mathcal{T}_t$

Then we have $\mathcal{U} \wedge \mathcal{T} \wedge I \wedge \phi \vdash_{tl} \bigcirc\phi$ iff $\mathcal{U}_t \wedge \mathcal{T}_t \wedge I(c) \wedge \phi(c) \vdash \phi(s(c))$. Here c is a newly introduced constant, substituted for the temporal parameter and $\vdash$ is meant to be first-order (non temporal) logic derivability.

In the next section we will apply this scheme to establish **at-most-one** properties for both the "Synapse N+1" and "Illinois" protocols.

6 Examples: the “Synapse N+1” and “Illinois” protocols

In this section we illustrate above approach by considering the verification of two cache coherence protocols, namely the “Synapse N+1” protocol and the “Illinois” protocol [5].

6.1 The “Synapse N+1” protocol

The “Synapse N+1” protocol [5] is a write-allocation cache coherence protocol with only three states, namely *invalid (I)*, *valid (V)* and *dirty (D)*, and two actions *read (R)* and *write (W)*. The transition relation is defined as follows:

$\tau(I,R) = V;\ \ \tau(I,W) = D;\ \ \tau(I,\bar{R}) = I;\ \ \tau(I,\bar{W}) = I;$
$\tau(V,R) = V;\ \ \tau(V,W) = D;\ \ \tau(V,\bar{R}) = V;\ \ \tau(V,\bar{W}) = I;$
$\tau(D,R) = D;\ \ \tau(D,W) = D;\ \ \tau(D,\bar{R}) = I;\ \ \tau(D,\bar{W}) = I;$

(Note that τ is a partial function.)

The initial condition is that all automata are in the same state, namely *invalid*. The correctness conditions are:

- the non-co-occurence condition for the states D and V; and
- the at-most-one condition for the state D.

6.1.1 Temporal Translation of the Protocol

First translate the above specification into a temporal specification. We have $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$, where

$\mathcal{U}$ **includes:**

$\exists x A(x);\ \ \forall x \forall y\ A(x) \wedge A(y) \rightarrow x = y;\ \ \forall x\ A(x) \leftrightarrow (R(x) \mid W(x));$
$\forall x\ (V(x) \mid I(x) \mid D(x));$

$\mathcal{S}$ **includes:** $\forall x\ I(x);$

$\mathcal{T}$ **includes:**

$\forall x\ ((I(x) \wedge \neg W(x) \wedge \exists y W(y)) \rightarrow \bigcirc I(x));\quad \forall x\ (I(x) \wedge W(x) \rightarrow \bigcirc D(x));$
$\forall x\ ((I(x) \wedge \neg R(x) \wedge \exists y\ R(y)) \rightarrow \bigcirc I(x));\quad \forall x\ (I(x) \wedge R(x) \rightarrow \bigcirc V(x));$
$\forall x\ ((V(x) \wedge \neg W(x) \wedge \exists y W(y)) \rightarrow \bigcirc I(x));\quad \forall x\ (V(x) \wedge W(x) \rightarrow \bigcirc D(x));$
$\forall x\ ((V(x) \wedge \neg R(x) \wedge \exists y R(y)) \rightarrow \bigcirc V(x));\quad \forall x\ (V(x) \wedge R(x) \rightarrow \bigcirc V(x));$
$\forall x\ ((D(x) \wedge \neg W(x) \wedge \exists y W(y)) \rightarrow \bigcirc I(x))\quad \forall x\ (D(x) \wedge W(x) \rightarrow \bigcirc D(x));$
$\forall x\ ((D(x) \wedge \neg R(x) \wedge \exists y\ R(y)) \rightarrow \bigcirc I(x));\quad \forall x\ (D(x) \wedge R(x) \rightarrow \bigcirc D(x)).$

Correctness Conditions: These are now defined as

χ_1**:** $\forall x \forall y\ (\neg D(x) \vee \neg V(y))$ [non-co-occurrence]

χ_2**:** $\forall x \forall y\ (D(x) \wedge D(y) \rightarrow x = y)$ [at-most-one]

Thus, the verification of the protocol is reduced to the following two ground induction problems: $SP \vdash \Box\chi_1$? and $SP \vdash \Box\chi_2$?

6.1.2 Monodic Specification

To translate the above temporal specification into a monodic one, we introduce 12 new predicate symbols ($I_1 - I_4$, $V_1 - V_4$, $D_1 - D_4$) and extend $\mathcal{U}$ to $\mathcal{U}'$ by the following definitions:

$$
\begin{array}{ll}
\forall x\ I_4(x) \leftrightarrow I(x) \wedge \neg W(x) \wedge \exists y\ W(y)); & \forall x\ I_2(x) \leftrightarrow I(x) \wedge W(x)); \\
\forall x\ I_3(x) \leftrightarrow I(x) \wedge \neg R(x) \wedge \exists y\ R(y); & \forall x\ I_1(x) \leftrightarrow I(x) \wedge R(x)); \\
\forall x\ V_4(x) \leftrightarrow V(x) \wedge \neg W(x) \wedge \exists y\ W(y)); & \forall x\ V_2(x) \leftrightarrow V(x) \wedge W(x); \\
\forall x\ V_3(x) \leftrightarrow V(x) \wedge \neg R(x) \wedge \exists y\ R(y))); & \forall x\ V_1(x) \leftrightarrow V(x) \wedge R(x); \\
\forall x\ D_4(x) \leftrightarrow D(x) \wedge \neg W(x) \wedge \exists y\ W(y)); & \forall x\ D_2(x) \leftrightarrow D(x) \wedge W(x); \\
\forall x\ D_3(x) \leftrightarrow D(x) \wedge \neg R(x) \wedge \exists y\ R(y)) & \forall x\ D_1(x) \leftrightarrow D(x) \wedge R(x).
\end{array}
$$

The set of step rules, $\mathcal{T}'$, in the new vocabulary is:

$$
\begin{array}{ll}
\forall x\ (I_1(x) \rightarrow \bigcirc V(x)); & \forall x\ (I_2(x) \rightarrow \bigcirc D(x)); \\
\forall x\ (I_3(x) \rightarrow \bigcirc I(x)); & \forall x\ (I_4(x) \rightarrow \bigcirc I(x)); \\
\forall x\ (V_1(x) \rightarrow \bigcirc V(x)); & \forall x\ (V_2(x) \rightarrow \bigcirc D(x)); \\
\forall x\ (V_3(x) \rightarrow \bigcirc V(x)); & \forall x\ (V_4(x) \rightarrow \bigcirc I(x)); \\
\forall x\ (D_1(x) \rightarrow \bigcirc D(x)); & \forall x\ (D_2(x) \rightarrow \bigcirc D(x)); \\
\forall x\ (D_3(x) \rightarrow \bigcirc I(x)); & \forall x\ (D_4(x) \rightarrow \bigcirc I(x))
\end{array}
$$

This gives $SP' = \langle \mathcal{U}', \mathcal{S}, \mathcal{T}' \rangle$ which is satisfiable if, and only if, SP is satisfiable.

6.1.3 Verification of the "non-co-occurence" Condition

To establish $SP' \vdash \chi_1$, we apply the method described in Section 5. Consider the following merged rules, generated from SP':

$$
\begin{array}{ll}
A_1 \rightarrow \bigcirc B_1\textbf{:} & \forall x(I_1(x) \vee I_3(x) \vee D_3(x)) \rightarrow \bigcirc \forall x\ (V(x) \vee I(x)) \\
A_2 \rightarrow \bigcirc B_2\textbf{:} & \forall x(I_2(x) \vee I_4(x) \vee D_4(x)) \rightarrow \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_3 \rightarrow \bigcirc B_3\textbf{:} & \forall x(D_1(x) \vee D_3(x) \vee I_3(x)) \rightarrow \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_4 \rightarrow \bigcirc B_4\textbf{:} & \forall x(D_2(x) \vee D_4(x) \vee I_2(x)) \rightarrow \bigcirc \forall x\ (D(x) \vee I(x))
\end{array}
$$

$$A_5 \to \bigcirc B_5\text{:}\ \ \forall x (V_1(x) \vee V_3(x) \vee I_3(x)) \to \bigcirc \forall x\ (V(x) \vee I(x))$$
$$A_6 \to \bigcirc B_6\text{:}\ \ \forall x (I_1(x) \vee I_3(x) \vee D_3(x)) \to \bigcirc \forall x\ (V(x) \vee I(x))$$
$$A_7 \to \bigcirc B_7\text{:}\ \ \forall x (I_1(x) \vee I_3(x) \vee V_3(x)) \to \bigcirc \forall x\ (V(x) \vee I(x))$$
$$A_8 \to \bigcirc B_8\text{:}\ \ \forall x (I_2(x) \vee I_4(x) \vee V_4(x)) \to \bigcirc \forall x\ (D(x) \vee I(x))$$

We have now:

- $\mathcal{U} \wedge B_i \vdash \chi_1$ for $i = 1 \ldots 8$
- $\mathcal{U} \wedge B_i \vdash \bigvee_{i=1}^{8} A_i$ for $i = 1 \ldots 8$
- $\mathcal{S} \wedge \mathcal{U} \vdash \chi_1 \wedge \bigvee_{i=1}^{8} A_i$

These items can be checked by any first-order theorem prover or, alternatively, by a decision procedure for (classical) monadic first-order logic. They are true and so it follows that $SP \to \Box\chi_1$ is a valid formula.

6.1.4 Verification of the "at-most-one" Condition

We use the induction scheme defined in Proposition 2 and as an invariant we take $\bigvee_{i=1}^{8} A_i$ as defined above.

Thus, we have to establish $\mathcal{U} \wedge \mathcal{T} \wedge \bigvee_{i=1}^{8} A_i \wedge \chi_2 \vdash_{tl} \bigcirc \chi_2$.

This is equivalent to showing $\mathcal{U} \wedge \mathcal{T} \wedge A_i \wedge \chi_2 \vdash_{tl} \bigcirc \chi_2$ for $i = 1, \ldots, 8$

Consider the case $i = 1$. Introducing the temporal parameter we have to show $\mathcal{U}_t \wedge \mathcal{T}_t \wedge A_1(c) \wedge \chi(c) \vdash \chi(s(c))$

Take the following formulae as a subset $\mathcal{T}'_t \subseteq tP_t$:

$$\forall t \forall x\ (I_1(x,t) \to V(x, s(t))),$$
$$\forall t \forall x\ (I_3(x,t) \to I(x, s(t))),$$
$$\forall t \forall x\ (D_3(x,t) \to I(x, s(t))).$$

Then, assuming $\mathcal{T}'_t, \mathcal{U}_t$, $A_i(c)$ and $\chi_2(c)$, one can demonstrate by straightforward first-order reasoning the validity of $\chi_2(s(c))$. All cases $i = 2, \ldots 8$ can be also demonstrated in a similar way.

So we can conclude that $SP \vdash \Box\chi_2$.

	I	D	S	V
$P \to Rm$	V	$-$	$-$	$-$
$\neg P \to Rm$	S	$-$	$-$	$-$
WI	D	$-$	D	$-$
Rh	$-$	D	S	V
We	$-$	$-$	$-$	D
Rm	$-$	$-$	$-$	$-$
Wd	$-$	D	$-$	$-$
Rep	$-$	I	I	I
$\bar{W}I$	I	I	I	I
$\bar{R}h$	I	D	S	V
$\bar{W}e$	I	D	S	V
$\bar{R}m$	I	S	S	S
$\bar{W}d$	I	D	S	V
$\bar{R}ep$	I	D	S	V

Table 1: Transition Relation of "Illinois" Protocol Example

6.2 The "Illinois" protocol

The "Illinois" protocol [13, 6] is an example of a cache coherence protocol using global conditions. This protocol uses four states, namely *Invalid (I)*, *Dirty(D)*, *Shared (S)*, *Valid Exclusive (V)*. Possible actions are *Read hit (Rh)*, *Write in exclusive state (We)*, *Read miss (Rm)* and *Write and Invalidate (WI)*, *Write in dirty state (Wd)* and *Replace (Rep)*.

The definition of the transition relation uses the following global conditions:

$P \equiv \#D = 0 \wedge \#S = 0 \wedge \#VE = 0$ and $\neg P \equiv \#D \geq 0 \wedge \#S \geq 0 \wedge \#VE \geq 0$

Table 1 presents the transition relation τ. Here, the cell in the column X (name of state) and the row Z (action) contains the name of the state Z iff $\langle X, Y, Z \rangle \in \tau$. If there is no state Z such that $\langle X, Y, Z \rangle \in \tau$ then the cell (X, Y) contains '$-'$.
The initial condition is that all automata are in the state *Invalid*. The correctness conditions are similar to those for Synapse N+1 algorithm:

- non-co-occurence condition for the states D and V; and
- at-most-one condition for the state D.

6.2.1 Temporal translation

First translate conditions P and $\neg P$ into logical form:

$$\Psi_P \equiv \forall x\ (\neg D(x) \wedge \neg S(x) \wedge \neg V(x)) \text{ and}$$
$$\Psi_{\neg P} \equiv \neg\Psi_P \equiv \exists x\ (D(x) \vee S(x) \vee V(x))$$

Next, translate the above specification of the protocol into a temporal specification. We have $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$, where

$\mathcal{U}$ includes:

$\exists x A(x)$;
$\forall x \forall y\ A(x) \wedge A(y) \rightarrow x = y$; $\forall x\ A(x) \wedge I(x) \leftrightarrow (Rm(x) \mid WI(x))$;
$\forall x; A(x) \wedge D(x) \leftrightarrow (Rh(x) \mid Wd(X) \mid Rep(x)$;
$\forall x\ A(x) \wedge S(x) \leftrightarrow (WI(x) \mid Rh(x) \mid Rep(x))$;
$\forall x\ A(x) \wedge V(x) \leftrightarrow (Rh(x) \mid We(x) \mid Rep(x))$;
$\forall x\ (V(x) \mid I(x) \mid D(x))$;

$\mathcal{S}$ includes: $\forall x\ I(x)$;

$\mathcal{T}$ includes:

$$\begin{aligned}
r_1 &: \forall x\ (I(x) \wedge Rm(x) \wedge \Psi_P \rightarrow \bigcirc V(x));\\
r_2 &: \forall x\ (I(x) \wedge Rm(x) \wedge \Psi_{\neg P} \rightarrow \bigcirc S(x));\\
r_3 &: \forall x\ (I(x) \wedge WI(x) \wedge \rightarrow \bigcirc D(x));\\
r_4 &: \forall x\ ((I(x) \wedge \neg WI(x) \wedge \exists y WI(y)) \rightarrow \bigcirc I(x));\\
r_5 &: \forall x\ ((I(x) \wedge \neg Rh(x) \wedge \exists y Rh(y)) \rightarrow \bigcirc I(x));\\
r_6 &: \forall x\ ((I(x) \wedge \neg We(x) \wedge \exists y We(y)) \rightarrow \bigcirc I(x));\\
r_7 &: \forall x\ ((I(x) \wedge \neg Rm(x) \wedge \exists y Rm(y)) \rightarrow \bigcirc I(x));\\
r_8 &: \forall x\ ((I(x) \wedge \neg Wd(x) \wedge \exists y Wd(y)) \rightarrow \bigcirc I(x));\\
r_9 &: \forall x\ ((I(x) \wedge \neg Rep(x) \wedge \exists y Rep(y)) \rightarrow \bigcirc I(x));
\end{aligned}$$

$$\begin{aligned}
r_{10} &: \forall x\ (D(x) \wedge Rh(x) \wedge \rightarrow \bigcirc D(x));\\
r_{11} &: \forall x\ (D(x) \wedge Wd(x) \wedge \rightarrow \bigcirc D(x));\\
r_{12} &: \forall x\ (D(x) \wedge Rep(x) \wedge \rightarrow \bigcirc I(x));\\
r_{13} &: \forall x\ ((D(x) \wedge \neg WI(x) \wedge \exists y WI(y)) \rightarrow \bigcirc I(x));\\
r_{14} &: \forall x\ ((D(x) \wedge \neg Rh(x) \wedge \exists y Rh(y)) \rightarrow \bigcirc D(x));\\
r_{15} &: \forall x\ ((D(x) \wedge \neg We(x) \wedge \exists y We(y)) \rightarrow \bigcirc D(x));\\
r_{16} &: \forall x\ ((D(x) \wedge \neg Rm(x) \wedge \exists y Rm(y)) \rightarrow \bigcirc S(x));\\
r_{17} &: \forall x\ ((D(x) \wedge \neg Wd(x) \wedge \exists y Wd(y)) \rightarrow \bigcirc D(x));\\
r_{18} &: \forall x\ ((D(x) \wedge \neg Rep(x) \wedge \exists y Rep(y)) \rightarrow \bigcirc D(x));
\end{aligned}$$

$r_{19}: \ \forall x\ (S(x) \wedge WI(x) \wedge \rightarrow \bigcirc D(x));$
$r_{21}: \ \forall x\ (S(x) \wedge Rh(x) \wedge \rightarrow \bigcirc S(x));$
$r_{22}: \ \forall x\ (S(x) \wedge Rep(x) \wedge \rightarrow \bigcirc I(x));$
$r_{23}: \ \forall x\ ((S(x) \wedge \neg WI(x) \wedge \exists y WI(y)) \rightarrow \bigcirc I(x));$
$r_{24}: \ \forall x\ ((S(x) \wedge \neg Rh(x) \wedge \exists y Rh(y)) \rightarrow \bigcirc S(x));$
$r_{25}: \ \forall x\ ((S(x) \wedge \neg We(x) \wedge \exists y We(y)) \rightarrow \bigcirc S(x));$
$r_{26}: \ \forall x\ ((S(x) \wedge \neg Rm(x) \wedge \exists y Rm(y)) \rightarrow \bigcirc S(x));$
$r_{27}: \ \forall x\ ((S(x) \wedge \neg Wd(x) \wedge \exists y Wd(y)) \rightarrow \bigcirc S(x));$
$r_{27}: \ \forall x\ ((S(x) \wedge \neg Rep(x) \wedge \exists y Rep(y)) \rightarrow \bigcirc S(x));$

$r_{28}: \ \forall x\ (V(x) \wedge Rh(x) \wedge \rightarrow \bigcirc V(x));$
$r_{29}: \ \forall x\ (V(x) \wedge We(x) \wedge \rightarrow \bigcirc D(x));$
$r_{30}: \ \forall x\ (V(x) \wedge Rep(x) \wedge \rightarrow \bigcirc I(x));$
$r_{31}: \ \forall x\ ((V(x) \wedge \neg WI(x) \wedge \exists y WI(y)) \rightarrow \bigcirc I(x));$
$r_{32}: \ \forall x\ ((V(x) \wedge \neg Rh(x) \wedge \exists y Rh(y)) \rightarrow \bigcirc V(x));$
$r_{33}: \ \forall x\ ((V(x) \wedge \neg We(x) \wedge \exists y We(y)) \rightarrow \bigcirc V(x));$
$r_{34}: \ \forall x\ ((V(x) \wedge \neg Rm(x) \wedge \exists y Rm(y)) \rightarrow \bigcirc S(x));$
$r_{35}: \ \forall x\ ((V(x) \wedge \neg Wd(x) \wedge \exists y Wd(y)) \rightarrow \bigcirc V(x));$
$r_{36}: \ \forall x\ ((V(x) \wedge \neg Rep(x) \wedge \exists y Rep(y)) \rightarrow \bigcirc V(x)).$

Correctness Conditions: the same as for the "Synapse N+1" protocol:

χ_1**:** $\forall x \forall y\ (\neg D(x) \vee \neg V(y))$

χ_2**:** $\forall x \forall y\ (D(x) \wedge D(y) \rightarrow x = y)$

Thus, the verification of the protocol is reduced to the following two ground induction problems: $SP \vdash \Box\chi_1$? and $SP \vdash \Box\chi_2$?

As in the case of "Synapse N+1" protocol we introduce new predicate symbols, one for the left hand side of each step formulae from $\mathcal{T}$: $I_1 - I_9$ for the rules $r_1 - r_9$; $D_1 - D_9$ for the rules $r_{10} - r_{18}$; $S_1 - S_9$ for $r_{19} - r_{27}$ and $V_1 - V_9$ for $r_{28} - r_{36}$ and extend $\mathcal{U}$ to $\mathcal{U}'$ by the corresponding definitions (as in the case of the Synapse N+1 protocol).

Then we get a modified set of step rules $\mathcal{T}'$:

$$\begin{array}{lll}
\forall x\ I_1(x) \to \bigcirc V(x); & \forall x\ I_2(x) \to \bigcirc S(x); & \forall x\ I_3(x) \to \bigcirc D(x); \\
\forall x\ I_4(x) \to \bigcirc I(x); & \forall x\ I_5(x) \to \bigcirc I(x); & \forall x\ I_6(x) \to \bigcirc I(x); \\
\forall x\ I_7(x) \to \bigcirc I(x); & \forall x\ I_8(x) \to \bigcirc I(x); & \forall x\ I_9(x) \to \bigcirc I(x); \\
\\
\forall x\ D_1(x) \to \bigcirc D(x); & \forall x\ D_2(x) \to \bigcirc D(x); & \forall x\ D_3(x) \bigcirc I(x); \\
\forall x\ D_4(x) \to \bigcirc I(x); & \forall x\ D_5(x) \to \bigcirc D(x); & \forall x\ D_6(x) \to \bigcirc D(x); \\
\forall x\ D_7(x) \to \bigcirc S(x); & \forall x\ D_8(x) \to \bigcirc D(x)); & \forall x\ D_9(x) \to \bigcirc D(x); \\
\\
\forall x\ S_1(x) \to \bigcirc D(x); & \forall x\ S_2(x) \to \bigcirc S(x); & \forall x\ S_3(x) \to \bigcirc I(x); \\
\forall x\ S_4(x) \to \bigcirc I(x); & \forall x\ S_5(x) \to \bigcirc S(x); & \forall x\ S_6(x) \to \bigcirc S(x); \\
\forall x\ S_7(x) \to \bigcirc S(x); & \forall x\ S_8(x) \to \bigcirc S(x); & \forall x\ S_9(x) \to \bigcirc S(x); \\
\\
\forall x\ V_1(x) \to \bigcirc V(x); & \forall x\ V_2(x) \to \bigcirc D(x); & \forall x\ V_3(x) \to \bigcirc I(x); \\
\forall x\ V_4(x) \to \bigcirc I(x); & \forall x\ V_5(x) \to \bigcirc V(x); & \forall x\ V_6(x) \to \bigcirc V(x); \\
\forall x\ V_7(x) \to \bigcirc S(x); & \forall x\ V_8(x) \to \bigcirc V(x); & \forall x\ V_9(x) \to \bigcirc V(x).
\end{array}$$

6.2.2 Verification of the "non-co-occurence" Condition

To establish $SP' \vdash \chi_1$, we again apply the method described in Section 5. Consider the following merged rules, generated from SP':

$$\begin{array}{ll}
A_1 \to \bigcirc B_1\textbf{:} & \forall x(I_1(x) \vee I_7(x)) \to \bigcirc \forall x\ (V(x) \vee I(x)) \\
A_2 \to \bigcirc B_2\textbf{:} & \forall x(V_1(x) \vee I_5(x)) \to \bigcirc \forall x\ (V(x) \vee I(x)) \\
A_3 \to \bigcirc B_3\textbf{:} & \forall x(V_2(x) \vee I_6(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_4 \to \bigcirc B_4\textbf{:} & \forall x(V_3(x) \vee I_9(x)) \to \bigcirc \forall x\ I(x) \\
A_5 \to \bigcirc B_5\textbf{:} & \forall x(V_4(x) \vee I_3(x)) \to \bigcirc \forall x\ (I(x) \vee D(x)) \\
A_6 \to \bigcirc B_6\textbf{:} & \forall x(V_7(x) \vee I_2(x)) \to \bigcirc \forall x\ S(x) \\
A_7 \to \bigcirc B_7\textbf{:} & \forall x(D_1(x) \vee I_5(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_8 \to \bigcirc B_8\textbf{:} & \forall x(D_2(x) \vee I_8(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_9 \to \bigcirc B_9\textbf{:} & \forall x(D_3(x) \vee I_9(x)) \to \bigcirc \forall x\ I(x) \\
A_{10} \to \bigcirc B_{10}\textbf{:} & \forall x(I_2(x) \vee D_7(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_{11} \to \bigcirc B_{11}\textbf{:} & \forall x(I_3(x) \vee D_4(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_{12} \to \bigcirc B_{12}\textbf{:} & \forall x(S_1(x) \vee S_4(x)) \to \bigcirc \forall x\ (D(x) \vee I(x)) \\
A_{13} \to \bigcirc B_{13}\textbf{:} & \forall x(S_2(x) \vee S_5(x)) \to \bigcirc \forall x\ S(x) \\
A_{14} \to \bigcirc B_{14}\textbf{:} & \forall x(S_3(x) \vee S_9(x)) \to \bigcirc \forall x\ (I(x) \vee S(x))
\end{array}$$

$$A_{15} \to \bigcirc B_{15}\text{:}\quad \forall x(I_2(x) \vee S_7(x)) \to \bigcirc\forall x\ S(x)$$
$$A_{16} \to \bigcirc B_{16}\text{:}\quad \forall x(I_3(x) \vee S_4(x)) \to \bigcirc\forall x\ (D(x) \vee I(x))$$
$$A_{17} \to \bigcirc B_{17}\text{:}\quad \forall x(I_4(x) \vee S_1(x)) \to \bigcirc\forall x\ (I(x) \vee D(x))$$
$$A_{18} \to \bigcirc B_{18}\text{:}\quad \forall x(I_5(x) \vee S_2(x)) \to \bigcirc\forall x\ (I(x) \vee S(x))$$
$$A_{19} \to \bigcirc B_{19}\text{:}\quad \forall x(I_9(x) \vee S_3(x)) \to \bigcirc\forall x\ I(x)$$

We have now:

- $\mathcal{U} \wedge B_i \vdash \chi_1$ for $i = 1 \ldots 19$
- $\mathcal{U} \wedge B_i \vdash \bigvee_{i=1}^{19} A_i$ for $i = 1 \ldots 19$. As an example: $\mathcal{U} \wedge B_1 \vdash \bigvee_{i=2}^{6} A_i$
- $\mathcal{S} \wedge \mathcal{U} \vdash \chi_1 \wedge \bigvee_{i=1}^{19} A_i$. In fact, we have $\mathcal{S} \wedge \mathcal{U} \vdash \chi_1 \wedge A_1$

Again, these items can be checked by any first-order theorem prover or, alternatively, by a decision procedure for (classical) monadic first-order logic. These are true and so it follows that $SP \to \Box\chi_1$ is a valid formula.

6.2.3 Verification of the "at-most-one" Condition

We use the induction scheme defined in Proposition 2 and as an invariant we take $\bigvee_{i=1}^{19} A_i$ as defined above. It is equivalent to showing $\mathcal{U} \wedge \mathcal{T} \wedge A_i \wedge \chi_2 \vdash_{tl} \bigcirc\chi_2$ for $i = 1, \ldots, 19$.

All 19 cases are translated then into first-order deductive problems, which can be resolved either by any first-order thorem prover, or alternatively by a decision procedure for (classical) monadic first-order logic. These are true and so it follows that $SP \to \Box\chi_2$ is valid formula.

7 Conclusion

We have demonstrated how methods developed for theorem proving in the monodic fragment of first-order temporal logic can be applied to the verification of parametrized cache coherence protocols. This is an important step, especially since it allows verification over arbitrary numbers of processes. It is also the first use of monodic FOTLs in the deductive verification of reactive systems.

Both theoretical and practical aspects need to be understood better. Theoretically, we would like to have a method that is provably complete for a large class of specifications. From the practical viewpoint we intend to adapt and optimize our implementation of the resolution based proof method [12] in order to verify all cache coherence protocols from [5] in pure automatic way. This would provide us with an answer to the question of whether our methods can compete with those based on reduction to arithmetical constraints [5].

References

[1] J. Brotherston, A. Degtyarev, M. Fisher and A. Lisitsa, Searching for Invariants using Temporal Resolution, in Proceeding of 9th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), Tbilisi, Georgia, 2002, LNAI 2514, Springer, pp 86–101

[2] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Dec. 1999.

[3] A. Degtyarev and M. Fisher. Towards first-order temporal resolution. In *Proccedings of KI-2001*, volume 2174 of *LNAI*, 2001.

[4] A. Degtyarev, M. Fisher and B. Konev. Monodic Temporal Resolution. *ACM Transactions on Computational Logic*, to appear, 2004.

[5] G. Delzanno. Automatic Verification of Parameterized Cache Coherence Protocols. In *Proc. Computer-Aided Verification (CAV)*, pp 53-68, LNCS 1855, Springer-Verlag, 2000.

[6] G. Delzanno. Constraint-based Verification of Parametrized Cache Coherence Protocols. to appear in Formal Methods in System Design.

[7] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computation Logic*, 2(1), January 2001.

[8] M. Fisher. A resolution method for temporal logic. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufman, 1991.

[9] M. Fisher. A normal form for temporal logics and its applications in theorem-proving and execution. *Journal of Logic and Computation*, 7(4), 1997.

[10] J. Handy. The Cache memory Book. ACademic Press, 1993.

[11] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Fragments of first-order temporal logics. *Annals of Pure and Applied logic*, 106:85–134, 2000.

[12] A. Lisitsa and J. Brotherston. Implementing Invariant search via Temporal resolution. In Proceedings of 2nd Workshop on the Workshop on Implementation of Logics, Tbilisi, Georgia, 2002.

[13] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1984.

[14] F. Pong and M. Dubois. A new approach for the Verfication of Cache Coherence Protocols. IEEE Transactions on Parallel and Distributed Systems 6(8), 1995.

[15] F. Pong and M. Dubois. Verification techniques for cache coherence protocols. ACM Computing Surveys 29(1): 82–126,1997.

[16] D.A. Plaisted and S.A. Greenbaum. A structure-preserving clause form transformation. *Journal of Symbolic Computation*, 2(3):293–304, September 1986.