Abstract Diagnosis for *tccp* using a Linear Temporal Logic *
Extended Abstract

M. Comini L. Titolo
Dipartimento di Matematica e Informatica, U. di Udine

A. Villanueva
DSIC, Universitat Politècnica de València

This extended abstract is a summary of [5], where we provided an automatic decision method to check whether a given property, specified in a linear temporal logic, is *valid* w.r.t. a *tccp* program. Our proposal (based on abstract interpretation techniques) does not require to build any model of the program, in constrast with standard verification methods such as model checking. Our results guarantee correctness but, as usual when using an abstract semantics, completeness is lost.

The Concurrent Constraint Paradigm (*ccp*, [8]) is a simple, logic model which is different from other (concurrent) programming paradigms mainly due to the notion of store-as-constraint that replaces the classical store-as-valuation model. It is based on an underlying constraint system that handles constraints on variables and deals with partial information. Within this family, [1] introduced the *Timed Concurrent Constraint Language* (*tccp* in short) by adding to the original *ccp* model the notion of time and the ability to capture the absence of information. With these features, one can specify behaviors typical of reactive systems such as *timeouts* or *preemption* actions.

It is well-known that modeling and verifying concurrent systems by hand can be an extremely hard task. Thus, the development of automatic formal methods is essential. One of the most known techniques for formal verification is model checking, that was originally introduced in [2, 7] to automatically check if a finite-state system satisfies a given property. It consisted in an exhaustive analysis of the state-space of the system; thus the state-explosion problem is its main drawback and, for this reason, many proposals in the literature try to mitigate it.

All the proposals of model checking have in common that a *part* of the model of the (target) program has to be built, and sometimes the needed fragment is quite huge. In [5], we propose a completely different approach to the formal verification of temporal (LTL) properties of concurrent (reactive) systems specified in *tccp*. We formalize a method to validate a specification of the expected behavior of a *tccp* program P, expressed by a linear temporal formula ϕ, which does not require to build any model at all.

The linear temporal logic we use to express specifications, csLTL, is an adaptation of the propositional LTL logic to the concurrent constraint framework. It is expressive enough to represent the abstract semantics of *tccp* with much precision. This logic is also used as the basis of the abstract domain for a new (abstract) semantics for the language.

In brief, our method is an *extension* of abstract diagnosis for *tccp* [3] where the abstract domain $\mathbb{F}$ is formed by csLTL formulas. Given a *tccp* program P and a specification $\phi \in$ csLTL that represent our intended behavior for P we are interested in decide if P is (abstractly) correct w.r.t. ϕ. We cannot use the original abstract diagnosis framework of [3] since $\mathbb{F}$ is not a complete lattice. We provide an automatic decision procedure for csLTL which, due to the underlying concurrent constraint model, has non trivial differences w.r.t. the classical propositional LTL.

The contributions of this work, published in [5], are the following:

*This work has been partially supported by the EU (FEDER) and the Spanish MINECO ref. TIN2013-45732-C4 (DAMAS), and by Generalitat Valenciana ref. PROMETEOII/2015/013.

Submitted to:
PROLE 2015

- A new abstract semantics for *tccp* programs based on csLTL formulas, which is correct w.r.t. the behavior of the language.
- A novel and effective method to validate csLTL properties based on the ideas of abstract diagnosis. This proposal intuitively consists in viewing P as a formula transformer by means of an (abstract) immediate consequence operator $\hat{\mathcal{D}}[\![P]\!]$ which works on csLTL formulas. Then, to decide the validity of ϕ, we just have to check if $\hat{\mathcal{D}}[\![P]\!]_\phi$ (i.e., the P-transformation of ϕ) logically implies ϕ;
- An automatic decision procedure for csLTL properties that makes our method effective. We have adapted to csLTL the tableau construction for Propositional LTL of [6]. [4] contains a preliminary version of the method.

  To check if a program P is abstractly correct w.r.t. a specification ϕ we just have to build the tableau for the negation of the formula $\hat{\mathcal{D}}[\![P]\!]_\phi \dot{\rightarrow} \phi$ and check if it is closed or not. If it is, we have that D is abstractly correct w.r.t. the specification ϕ. Otherwise, we can extract from the open branch of the tableau an explicit testimony of the abstract incorrectness of P.

With our technique we can check, for instance, that, at a railway crossing system, each time a train is approaching, the gate is down, or that whenever a train has crossed, the gate is up. When a property is non valid, the method identifies the buggy process declaration.

When applying the diagnosis w.r.t. approximate properties, the results may be weaker than those that can be achieved on concrete domains just because of approximation. Abstract incorrect process declarations are in general just a warning about a possible source of errors. Because of the approximation, it can happen that a (concretely) correct declaration is abstractly incorrect. However, all concrete errors are detected.

References

[1] F. S. de Boer, M. Gabbrielli & M. C. Meo (2000): *A Timed Concurrent Constraint Language*. *Information and Computation* 161(1), pp. 45–83. (document)

[2] E. M. Clarke & E. A. Emerson (1982): *Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic*. In: *Workshop on Logic of Programs, LNCS* 131, Springer, pp. 52–71. (document)

[3] M. Comini, L. Titolo & A. Villanueva (2011): *Abstract Diagnosis for Timed Concurrent Constraint programs*. *Theory and Practice of Logic Programming* 11(4-5), pp. 487–502. (document)

[4] M. Comini, L. Titolo & A. Villanueva (2013): *Towards an Effective Decision Procedure for LTL formulas with Constraints*. In: *23rd Workshop on Logic-based methods in Programming Environments (WLPE 2013)*. Available at http://arxiv.org/abs/1308.4171. (document)

[5] M. Comini, L. Titolo & A. Villanueva (2014): *Abstract Diagnosis for tccp using a Linear Temporal Logic*. *Theory and Practice of Logic Programming* 14(4-5), pp. 487–502. (document)

[6] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro & F. Orejas (2009): *Dual Systems of Tableaux and Sequents for PLTL*. *The Journal of Logic and Algebraic Programming* 78(8), pp. 701–722. (document)

[7] J. P. Queille & J. Sifakis (1982): *Specification and verification of concurrent systems in CESAR*. In: *Proc. of International Symposium on Programming, 5th Colloquium, LNCS* 137, Springer, pp. 337–351. (document)

[8] V. A. Saraswat (1993): *Concurrent Constraint Programming*. The MIT Press, Cambridge, Mass. (document)