Incorporating Homomorphic Reduction into Abstract Interpretation

Vassilios A. Zoukos[1,2], G. Papakonstantinou[1], and P. Tsanakas[1]

[1] Elec. & Comp. Eng. Dept.
National Technical University Athens,
Zographou Campus, GR-15770
Athens, Greece.
`e-mail: {zoukos,papakon,panag}@softlab.ece.ntua.gr`

[2] Intracom S.A.
Access & Transmission Dept.
Markopoulou Ave. 19002 Peania, Greece.
`e-mail: vzou@intracom.gr`

Abstract. It is well known that a major obstacle to model-checking is the *state-space explosion* problem. To overcome this problem, various techniques for systems' *abstractions* have been proposed. While these techniques seem to be unrelated, they share a common background: the general theory of *abstract interpretation.* In the field of automata-theoretic formal verification theory, *homomorphic reduction* has been proposed as an abstraction technique to make the verification process tractable. We show that homomorphic reduction is a special application of abstract interpretation to trace semantics of a system comprised of coordinating processes.

Key words: Model-Checking; Abstract Interpretation; Homomorphic Reduction.

1 Introduction

This report, comes out as a result of a study aimed to the foundation of a concrete theoretical background behind the *abstraction* of a transition system.

Presently, the quest for application of *abstraction techniques* to various transition systems, has been done a rapidly expanding field of research. The underlying motivation of this research is system's *formal verification.*

Without going into details, formal verification of an industrial size transition system, faces a major obstacle: the so-called *state-space explosion* problem. After years of research and experience, it has become now widely acceptable that the every promising technique for alleviating this problem, it is likely to be based on *abstractions* of system's model.

Roughly speaking, with the term "abstraction" of a system's model we mean the creation of an "approximate" model in such a way that the set of its behaviors is a superset of the "concrete" behaviors.

History The idea of the abstraction for a transition system exists in different application areas of computer science. It first appearance – to our best knowledge – is at the field of compilers construction. Its application aims to the a-priori determination of programs run-time properties under the general term *static program analysis.* This technique in essence, uses computers to detect program faults.

The application of this technique – to program computers such that they can analyze the programs that will execute – is a very difficult task. This difficulty comes from undecidability and complexity problems.

To overcome this difficulty, someone has to make reasonable compromises which basically constitute "approximations" of system's possible behaviors. Such an approximate method is the *abstract interpretation.* Its first public appearance is in 1977 in the publication of Cousot,Cousot [1].

During the 80's, the research on the formal verification/analysis of communication protocols, faced again with the problem of handling the state-space explosion produced from the parallel composition of concurrent programs. The solution proposed to get around this problem was to conduct the verification on "approximate" system models with tractable state space through *reduction mappings.* This technique appeared in Kurshan's publication [6] in 1987 under the term *homomorphic reduction* accompanied with a practical implementation in his formal verification tool COSPAN.

While the techniques of abstract interpretation and homomorphic reduction evolved through different origins, and up to now do not refer each other, they share the same mathematical background. We will prove in this paper, that the approximation of transition systems through reduction mappings, is in fact a special case of abstract interpretation applied to labeled transition systems.

Organization The rest of the paper is organized as follows:

In Section 2. we review the mathematical preliminaries and describe in a semi-formal way the basic concepts related to formal verification of transition systems, homomorphic reductions and abstract interpretation.

In Section 3. we give the formal definition of the *Galois connection* and state its most useful properties. We conclude this section with the properties of a special case of Galois connections: those between two Boolean algebras.

Finally, we relate these special Galois connections between Boolean algebras with Kurshan's duality between language homomorphisms and Boolean algebras homomorphisms. Due to space limitations we do not present the proofs for the stated propositions[1]

2 Preliminaries, Basic Concepts

In the rest of this paper we will use the following notation and terminology. A *poset* is the pair $(P, \sqsubseteq)$ where $\sqsubseteq$ is a partial order. With $x \sqsubset y$ we mean $x \sqsubseteq y$ and $x \neq y$. We say that y *covers* x in the partial order $\sqsubseteq$ iff $x \sqsubseteq y$, $x \neq y$ and $x \sqsubseteq z, z \sqsubset y \Rightarrow z = x$. If $X \subseteq P$, then the *down-set* denoted with $\downarrow Y$ is the set $\{y \in P \mid (\exists x \in X) : y \sqsubseteq x\}$

[1] These proofs are available by request at: `zoukos@softlab.ece.ntua.gr`.

while the *up-set* is the $\uparrow X = \{y \in P \mid (\exists x \in X) : y \sqsupseteq x\}$. We use $\sqcup X$, and $\sqcap X$, for the *least upper bound* (`lub`) of X in P, and the *greatest lower bound* (`glb`) of X in P respectively if these exist. The *minimum* (resp. *maximum*) element of the poset P is denoted with $\perp$ (resp. $\top$).

A poset $(L, \sqsubseteq)$ is a *lattice* iff for any $x, y \in L$ the $\sqcup\{x, y\}$ and $\sqcap\{x, y\}$ exist. A lattice $(L, \sqsubseteq)$ is *complete* if $\sqcup X$ and $\sqcap X$ exist for every $X \subseteq L$. If L is a lattice with $\perp$, then the set $\mathcal{A}(L)$ of its *atoms* has as members the elements of L covering $\perp$.

A lattice is *distributive* iff: $(\forall x, y, z \in L)\ x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Suppose that L is a lattice with $\perp, \top$. The *complement* of $x \in L$ is the element y such that: $x \sqcap y = \perp$ and $x \sqcup y = \top$. If x has a unique complement we denote it with $\sim x$.

Definition 2.1 [3, §7.2] A lattice L is a *Boolean lattice* or a *Boolean algebra* iff:

1. L is a distributive.
2. L has $\top$, $\perp$.
3. Every element $x \in L$ has a unique complement $\sim x \in L$.

For the Boolean algebras we use a slight modified notation: We denote with $+$, $*$ the operations $\sqcup$ and $\sqcap$, and with 0, 1 the elements $\perp$ and $\top$ respectively.

Every finite Boolean algebra is *atomic* (i.e. the set $\mathcal{A}(L)$ exists and every element $x \in L$ can be expressed as the lub of a subset of $\mathcal{A}(L)$). In this paper, when we refer to Boolean algebra, we mean a finite one.

Definition 2.2 [3, §7.4] Given a Boolean algebra L and a subset K of L, K is a *subalgebra* of L if it is closed with respect to $+, *, \sim$ and $\{0, 1\} \subseteq K \cap L$.

Definition 2.3 [5, §2.1.7] For the Boolean algebras L, L' the mapping :

$$h : L \to L'$$

is a *homomorphism* iff:

$$\begin{aligned} h(x + y) &= h(x) + h(y) \\ h(x * y) &= h(x) * h(y) \\ h(\sim x) &= \sim h(x). \end{aligned}$$

A *monomorphism* is an 1-1 homomorphism, while an *isomorphism* is a monomorphism onto. In the case where h is isomorphism we write $L \cong L'$. If $L = L'$ and h isomorphism, we say that h is an *automorphism.*

If L is a Boolean algebra and L_i a subalgebra of L, we define the *projection* of L to L_i the mapping:

$$\Pi_{L_i} : L \to L_i, \quad \Pi_{L_i}(x) = \sqcup(\uparrow(\downarrow\{x\} \cap \mathcal{A}(L)) \cap \mathcal{A}(L_i)).$$

Labeled Transition Systems A possible model for *discrete dynamic systems*, is the model of *labeled transition systems* (LTS) where the system's transitions are labeled by a subset of the "observable" events. In a formal way, we can say that the semantic basis of LTS are edge-labeled directed graphs where the labels are elements of an atomic Boolean algebra L. Such an approach to model systems of coordinating processes is taken by Kurshan (cf. [5, Ch. 3]) in his automata-theoretic verification theory.

Formal Verification With the term "formal verification" of a program we mean mathematical proofs about program's behaviors. While in general, formal verification can be viewed as theorem-proving in a given logic, practically spans a spectrum in which expressive power trades off against automatability.

Kurshan's approach to formal verification has as semantic basis the ω-automata. In this framework, –which is a form of *model checking* – the system is modelled by an automaton P, and the "property" or specification to be checked by the automaton T, and the test is whether the set of the "behaviors" of P is contained in the set of the behaviors of T. In formal notation:

$$\mathcal{L}(P) \subseteq \mathcal{L}(T).$$

This test is called *language containment* test and the sets $\mathcal{L}(P), \mathcal{L}(T)$ are subsets of $(\Sigma^{\natural})^{\omega}$ (the set of infinite sequences of $\Sigma^{\natural}$, where $\Sigma^{\natural}$ is the input/output alphabet of the system).

The language containment test can be done either by explicit state enumeration algorithms either by implicit (BDD-based) algorithms. In both cases the check requires an exhaustive search of the reachable state space of the system. Since P is usually expressed as a set of coordinating components $P = \otimes P_i$, the search state space generally increases exponentially to the number of its components (this is the *state-space* explosion problem).

Homomorphic Reduction One technique to get around state-space explosion, during the language containment test, is *homomorphic reduction.* Both the system P and the property T are mapped to the "abstract" system P' and property T' respectively, through a *language homomorphism* $\Phi : (\Sigma^{\natural})^{\omega} \rightarrow (\Sigma^{\sharp})^{\omega}$ (here $\Sigma^{\natural}$ is the "concrete" alphabet and $\Sigma^{\sharp}$ is the "abstract" one) and the language containment check is performed to the abstract models:

$$\mathcal{L}(P') \subseteq \mathcal{L}(T').$$

In the case where the mapping Φ satisfies[2] certain conditions between the languages of P, P', T, T', then the validity of the language containment test in the abstract models, implies the validity of the language containment test in the concrete models. In this case, we say that the abstract models P', T' constitute a *homomorphic reduction* of P, T.

The mapping Φ is defined as follows:

Let the Boolean algebras $L^{\natural} \cong \mathcal{P}(\Sigma^{\natural}), L^{\sharp} \cong \mathcal{P}(\Sigma^{\sharp})$ where $\Sigma^{\natural}, \Sigma^{\sharp}$, are the input/output alphabets of the concrete and abstract models respectively,

$$\hat{h} : \mathcal{A}(L^{\natural}) \rightarrow \mathcal{A}(L^{\sharp}),$$

an arbitrary map and define:

$$\Phi : \mathcal{A}(L^{\natural})^{\omega} \rightarrow \mathcal{A}(L^{\sharp})^{\omega},$$

[2] For a formal exposition on the subject see [5, §8.5]

by $\Phi((x_0,\ldots)) = (\hat{h}(x_0),\ldots)$. Then Φ is said to be a *language homomorphism* with *support* $\hat{h}$. Φ also extended naturally to ω-languages: subsets of $\mathcal{A}(L)^\omega$.

Note that the mapping Φ "abstracts" the input/output sequences of the concrete model. This is a case of trace semantics approximation.

The support map $\hat{h}$ is related to a unique Boolean algebra homomorphism h as we will see next.

Abstract Interpretation We will informally describe the basic ideas behind abstract interpretation. For a formal treatment see [2],[1].

Abstract interpretation is a method for designing approximate semantics of programs which can be used to gather information about programs in order to provide sound answers to questions about their run-time behaviors.

Theoretically, the purpose of abstract interpretation is to construct hierarchies of semantics specifying at different levels of abstraction the behavior of programs. In practice, abstract interpretation, helps in the implementation of tools for automatic analysis statically dynamic properties of programs.

The abstract interpretation mathematical framework is based on the use of *Galois connections* to establish the relationship between the domain of "concrete" or "exact" properties to the domain of "abstract" or "approximate" properties.

The intuition is that the abstract domain is a representation of some approximate properties of the values of the concrete domain. Both on the concrete and on the abstract domain, a partial order relation describing the relative precision of the values is defined: $x \sqsubseteq y$ means that x is more precise than y.

3 Galois Connections

In the next, we put the formal definitions of Galois connection and insertion.

Definition 3.1 The pair of mappings $\langle\alpha,\gamma\rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$, between the posets $(L^\natural, \sqsubseteq), (L^\sharp, \sqsubseteq)$, is a *Galois connection*[3] iff one of the following two equivalence conditions hold:

1. $p \sqsubseteq \gamma(q) \Leftrightarrow \alpha(p) \sqsubseteq q$.
2. $\begin{array}{lll} (i)\ p \sqsubseteq p' \Rightarrow \alpha(p) \sqsubseteq \alpha(p') & \wedge\ q \sqsubseteq q' \Rightarrow \gamma(q) \sqsubseteq \gamma(q'), & p,p' \in L^\natural, q,q' \in L^\sharp, \\ (ii)\ p \sqsubseteq \gamma \circ \alpha(p) & \wedge\ \alpha \circ \gamma(q) \sqsubseteq q, & p \in L^\natural, q \in L^\sharp. \end{array}$

In a Galois connection $\langle\alpha,\gamma\rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$, α and γ constitute its *abstraction mapping* and *concretization mapping* respectively. In the rest of the paper, for any pair of functions $\langle f, g\rangle$ of a Galois connection the following convention is adapted: f will be the abstraction mapping and g the concretization mapping.

Definition 3.2 A Galois connection $\langle\alpha,\gamma\rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$, is an *insertion* iff:

$$(\forall q \in L^\sharp)\ q = \alpha \circ \gamma(q).$$

[3] Also called and *semi-dual Galois correspondence*

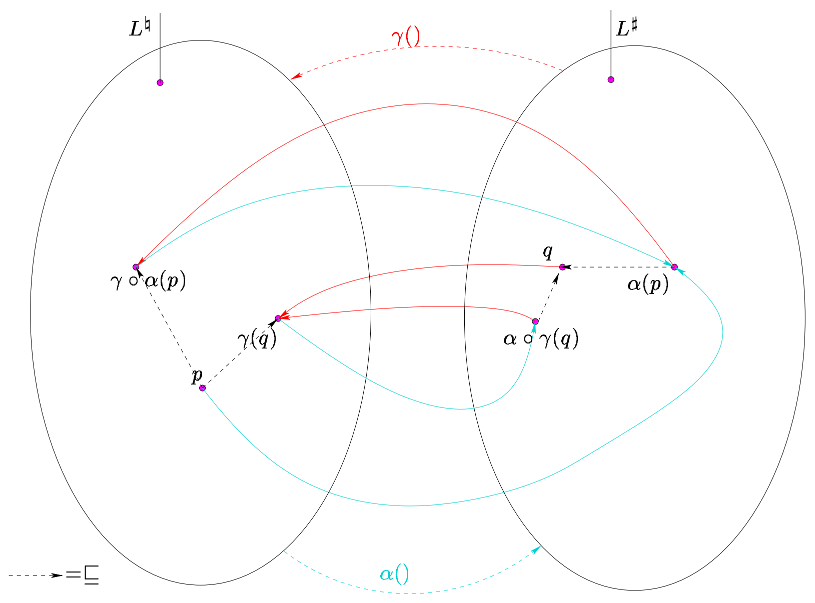

Fig. 1. A graphical representation of a Galois connection $\langle \alpha, \gamma \rangle$ between $L^\natural, L^\sharp$.

The following are some well-known properties of Galois connections and insertions that will be useful later on.

Proposition 3.1 *Let the Galois insertion:* $\langle \alpha, \gamma \rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$. *Where* $L^\natural, L^\sharp$ *are posets. Then:*

1. *The mapping* γ *is a monomorphism (or 1-1).*
2. *The mapping* α *is onto.*
3. *The mapping* γ *is an* embedding *of* $(L^\sharp, \sqsubseteq)$ *into* $(L^\natural, \sqsubseteq)$. *(i.e.* $(\forall q, q' \in L^\sharp)\ q \sqsubseteq q' \Leftrightarrow \gamma(q) \sqsubseteq \gamma(q')$.*)*

Proposition 3.2 *[3, Ch. 11 Ex.11.3(iv)] Let:* $\langle \alpha, \gamma \rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$ *be a Galois connection where* $L^\natural, L^\sharp$ *are posets. Then:*

1. α *preserves the* `lub`*'s. (i.e. if for* $X \subseteq L^\natural$ *the* $\sqcup X$ *exists, then* $\sqcup \alpha[X]$ *exists in* $L^\sharp$ *and* $\alpha(\sqcup X) = \sqcup \alpha[X]$.*)*
2. γ *preserves the* `glb`*'s. (i.e. if for* $Y \subseteq L^\sharp$ *the* $\sqcap Y$ *exists, then* $\sqcap \gamma[Y]$ *exists in* $L^\natural$ *and* $\gamma(\sqcap Y) = \sqcap \gamma[Y]$.*)*

Proposition 3.3 *[4, §2.1 (iii)] Let the Galois insertion:* $\langle \alpha, \gamma \rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$. *Where* $(L^\natural, \sqsubseteq)$ *is a complete lattice and* $(L^\sharp, \sqsubseteq)$ *is a poset. Then* $(L^\sharp, \sqsubseteq)$ *is a complete lattice.*

Proposition 3.4 *[4, §2.1 (iv)] Let the posets: $(L^\natural, \sqsubseteq), (L^\sharp, \sqsubseteq)$ and the mapping $\alpha \in L^\natural \to L^\sharp$ which preserves the* `lub`*'s. In addition, for all $y \in L^\sharp$ we assume that $\sqcup\{x \in L^\natural \mid \alpha(x) \sqsubseteq y\}$ exists. If we define: $\gamma \in L^\sharp \to L^\natural$ as $\gamma(y) = \sqcup\{x \in L^\natural \mid \alpha(x) \sqsubseteq y\}$. Then the pair $\langle\alpha, \gamma\rangle$ is a Galois connection. Moreover, if α is onto, then $\langle\alpha, \gamma\rangle$ is a Galois insertion.*

Proposition 3.5 *[4, §2.1 (v)] In any Galois connection, one of two functions uniquely determines each other as follows:*

$$\begin{aligned}\alpha(x) &= \sqcap\gamma^{-1}[\uparrow\{x\} \cap \gamma[L^\sharp]]\\ \gamma(y) &= \sqcup\alpha^{-1}[\downarrow\{y\} \cap \alpha[L^\natural]]\end{aligned}$$

Galois Connections between Boolean Algebras

When the posets $L^\natural, L^\sharp$ are Boolean Algebras, we can construct special types of Galois connections where the concretization function is a homomorphism. Proposition 3.6 states the necessary conditions for this.

Proposition 3.6 *Given two Boolean algebras $L^\natural, L^\sharp$ and a mapping $\hat{h} : L^\natural \to L^\sharp$ preserving the* `lub`*'s such that: $\hat{h}[\mathcal{A}(L^\natural)] \subseteq \mathcal{A}(L^\sharp)$, then the pair $\langle\hat{h}, h\rangle$ where $h(x) = \sqcup\hat{h}^{-1}[\downarrow\{x\} \cap \hat{h}[L^\natural]]$, is a Galois connection and moreover, the function h is a homomorphism. If $\hat{h}[\mathcal{A}(L^\natural)] = \mathcal{A}(L^\sharp)$, then h is a monomorphism.*

Proposition 3.7 *Given a Galois connection $\langle\hat{h}, h\rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$ where $L^\natural, L^\sharp$ are Boolean algebras, such that the concretization function h is a homomorphism, then the abstraction function $\hat{h}$ has the following properties:*

1. $\hat{h}(0) = 0$.
2. $\hat{h}(x + y) = \hat{h}(x) + \hat{h}(y)$.
3. $x \sqsubseteq y \Rightarrow \hat{h}(x) \sqsubseteq \hat{h}(y)$.
4. $\hat{h}(x * y) \sqsubseteq \hat{h}(x) * \hat{h}(y)$.

Proposition 3.8 *Let the Galois connection $\langle\hat{h}, h\rangle \in (L^\natural \to L^\sharp) \times (L^\sharp \to L^\natural)$ where $L^\natural, L^\sharp$ are Boolean algebras, and the concretization function h is a monomorphism. Then in addition to (1, 2, 3, 4) of the Proposition 3.7, the abstraction function $\hat{h}$ has the the following properties:*

1. $\hat{h}(1) = 1$.
2. $\sim \hat{h}(x) \sqsubseteq \hat{h}(\sim x)$.
3. $\hat{h} \circ h = id$.

The abstraction function $\hat{h}$ in the case of Proposition 3.8, is a *projection* $\Pi_{L'} : L^\natural \to L'$ where $L' = h[L^\sharp]$ the Boolean subalgebra of $L^\natural$ isomorphic to $L^\sharp$.

4 Discussion

The language homomorphism Φ is defined through its support mapping $\hat{h} : \mathcal{A}(L^{\natural}) \rightarrow \mathcal{A}(L^{\sharp})$. If we extent linearly $\hat{h}$ to $L^{\natural}$ as $\hat{h} : L^{\natural} \rightarrow L^{\sharp}$ then $\hat{h}$ satisfies all the conditions of Proposition 3.6 to be the abstraction mapping of a Galois connection $\langle \hat{h}, h \rangle$ between the Boolean algebras $L^{\natural}, L^{\sharp}$. In this case the concretization mapping h is a homomorphism. This remark, forms the mathematical basis for the theory of the duality between language homomorphisms and Boolean algebra homomorphisms and puts homomorphic reduction inside the framework of abstract interpretation.

Comparing the results in the previous section with Kurshan's development in [5, Ch.4], we can easily see that Proposition 3.6 is a restatement of Corollary 4.2.15 and Theorem 4.2.9. Proposition 3.7 combines the results of Lemmas 4.2.4 and 4.2.6, and Proposition 3.7 rephrases Theorem 4.2.7.

Homomorphic reduction in essence is an approximation of system's *trace semantics* through a language homomorphism. Such an approximation depends on the abstract alphabet $\Sigma^{\sharp}$ chosen each time.

While the incorporation of homomorphic reduction inside the framework of abstract interpretation it is rather a "theoretical" task, it might have and practical consequences too.

For example, current tools for formal verification of labeled transition systems such as COSPAN, can be greatly benefited by the incorporation of static analysis techniques in order to provide wider applicable results in industrial size application of formal verification.

References

1. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.
2. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, (aug) 1992.
3. B. A. Davey and H. A. Priestley. *Intruction to Lattices and Order.* Cambridge University Press, 1990.
4. Gilberto Filé and Francesco Ranzato. The powerset operator on abstract interpretations. *Theoretical Computer Science*, 222:77–111, 1999.
5. R. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach.* Princeton Univ. Press, 1994.
6. R. P. Kurshan. Reducibility in analysis of coordination. *LNCIS*, 103:19–39, 1989.