# Abstraction techniques for verification of concurrent systems

Constantin Enea

# Abstraction Techniques for Verification of Concurrent Systems

–Ph.D. Thesis –

by Constantin Enea

Supervisors:

Prof.dr. Ferucio Laurentiu Tiplea
Departement of Computer Science
"Al. I. Cuza" University of Iasi
Iasi, Romania

Prof.dr. Anatol Slissenko
Laboratoire d'Algorithmique, Complexite et Logique
University Paris 12, Val de Marne
Paris, France

Paris, 2007

# Preface

Abstraction techniques, often based on abstract interpretation, provide a method for symbolically executing systems using the abstract instead of the concrete domain. In this thesis, we are concerned with abstractions for logics under multi-valued interpretations. We provide preservation results for first-order logic, temporal logic, and temporal logic of knowledge. As a case study, we show how abstraction can be used to solve the safety problem for protection systems which model access control policies.

The use of abstraction in the context of data types, is also investigated. This technique scales well from data types to abstract data types. Here, abstractions are applied to initial specifications by means of equations and they are called equationally specified abstractions. To reason about dynamic systems, we introduce dynamic data types and extend the previous abstraction technique to this case.

The main problem that arises when using abstraction techniques is to find the suitable abstraction or to refine an already existing abstraction in order to obtain a better one. In this thesis, we prove that the abstraction techniques for data types, under Kleene's three-valued interpretation, can be used in a refinement procedure. Moreover, we show that the counterexample guided abstraction refinement procedure (CEGAR) [25] works better when used with equationally specified abstractions.

I would like to thank all the people that have contributed to this thesis in one way or another. First of all, there is Ferucio Laurentiu Tiplea, my supervisor from Romania, teacher, and co-author, who has always been available to answer my questions. His broad perspective of computer science, and far beyond, has helped me many times to make the right choices – but he has never forced anything upon me. He started working with me even from the undergraduate studies, making my life easier during the PhD.

My thanks should also be given to Anatol Slissenko and Catalin Dima, my supervisors at University "Paris 12". Their encouragement, guidance and feedback has been particularly important.

Last but not least, I would like to thanks my family and my friends for

their support and encouragement during the last years. Special thanks go to my parents to whom I owe most of what I know and what I am, and to Cezara for her support that helped me to get through the rough times as well as for being part of the good times.

September 15, 2007
Constantin Enea

# Contents

# Introduction

As the hardware and software systems are growing continuously in scale and functionality, the likelihood of subtle errors becomes greater. In industry, testing has traditionally been the main debugging technique. For example, "beta-releases" of a software are sent out to a group of people who are willing to use it and report on errors encountered. Other programs, like microprocessor code implemented in hardware, are often automatically tested by feeding them sequences of inputs and comparing the corresponding outputs to the desired ones. The validity of the testing approach is based on the exhaustive search for possible inputs (which can be impracticably, many) and it does not apply to *reactive systems* that maintain a continuous interaction with their environment. The reactive systems are in contrast to the old-fashioned view of a program as something that takes some input, computes for a while, and then produces a result and terminates.

The use of *formal methods* have been proposed to overcome these problems. Formal methods cover all approaches to specification and verification based on mathematical formalisms. They contain three basic parts: a mathematical model of the system, a formal language for expressing the specification and a methodology to establish whether the model of the system satisfies the specification.

Formal methods can be classified as *syntactic* and *semantic*. In syntactic methods the system is described in some programming language whose elementary constructs are expressed by axioms and the larger constructs by inference rules in some proof system. The specification is given in some powerful formal language and the proof of correctness reduces to a proof within this system. In semantic methods, the model of a program consists of a description of all its possible behaviors in a mathematical structure like a transition system. The specification is a formula in a logic that is interpreted over such structures (e.g. temporal logic) and the correctness is proved by showing that the formula is satisfied by the model.

An example of a semantic formal method is the *temporal logic model checking* developed independently by Clarke and Emerson [22] in the United

States and by Quielle and Sifakis [101] in France. In this approach, systems are modeled by Kripke structures and the specifications are expressed in a propositional temporal logic. An efficient search procedure is used to determine if the specification is true in the Kripke structure. The most important advantages of model checking are that it is completely automatic and in case of "false" answers it provides a counterexample that shows why the specification is not satisfied.

The main drawback of model checking is the *state explosion* problem that can occur in systems consisting of a set of concurrent processes which run in parallel. The set of possible behaviors of the system contains a sequence for each possible interleaving of actions of the components and it may grow exponentially in the number of components. The presence of data values can also contribute to the problem. An extra bit of memory used by a program potentially doubles the size of the state space. Many possible solutions to the state explosion problem have been proposed. They try to improve the representation of the full state space or to reduce the state space by ignoring certain details. In the first category we can find *symbolic model checking* [14, 85] which represents the state space by ordered binary decision diagrams [12] and produced spectacular results [14, 24, 88], or *on-the-fly model checking* [32, 50] which expands progressively the state space of the system. From the second category, we can mention *partial order* techniques [56, 99, 112], *symmetry* techniques [23, 43, 68], *modularization* [29, 58, 59], or *abstraction* [33, 69, 70, 77, 67, 15, 110, 100, 26, 114, 34, 57, 103, 36, 86, 4, 113, 104].

Abstraction techniques, often based on abstract interpretation [33], provide a method for symbolically executing systems using the abstract instead of the concrete domain. For example, the *data abstraction* technique from [26] considers an abstraction mapping between the actual data values in the system and a small set of abstract data values. This mapping is extended to states and transitions, in order to obtain an abstract version of the system under consideration. *Shape analysis* [96, 104], which is a data flow analysis technique used mainly for complex analysis of dynamically allocated data, is based on representing the set of possible memory states ("stores") that arise at a given point in the program by *shape graphs*. In such a graph, heap cells are represented by shape-graph nodes and, in particular, sets of "indistinguishable" heap cells are represented by a single shape-graph node. *Predicate abstraction* is another prominent abstraction technique [57, 36, 113, 4]. The main idea of predicate abstraction is to map concrete objects (states of a transition system, data of a data type etc.) to "abstract objects" according to their evaluation under a finite set of predicates. In [6], Bidoit and Boisseau consider *algebraic abstractions* in order to verify properties of security protocols modeled by universal algebras. Their abstraction is based on ho-

momorphisms, and the technique of *duplicating predicate symbols* [26, 35] is used for validation and refutation.

In order that an abstraction be useful, it must be *property-preserving*. The forms of property preservation which are mostly studied in the literature involve only logics under the classical two-valued interpretation. Multi-valued logics provide an alternative to classical boolean logic for modeling and reasoning about systems. By adding new truth values, uncertainty and disagreement can be modeled explicitly and a variety of applications were found in databases [53], knowledge representation [55], machine learning [93], software and hardware verification [9, 10, 65].

Many applications of multi-valued logics have been found in hardware and software verification. For hardware verification, simulation tools and implementations of genuinely multi-valued circuits have been proposed, dynamic hazards have been modeled by introducing pseudo states to find overlapping regions of competing signals [13], implementation of gates have been verified on the basis of switch level models [64], etc. For software verification, we need uncertainty because we may not know whether some behaviors should be possible, we need disagreement because we may have different stakeholders that disagree about how the systems should behave and we need to represent relative importance because some behaviors are essential and others may or may not be implemented. *Multi-valued model checking* techniques have been proposed by many researchers [16, 72, 19, 17, 20, 60, 11, 73, 74], thus, motivating, even more, the use of multi-valued logics in the verification process.

This thesis deals with abstractions that preserve properties expressed in logics under multi-valued interpretations. Chapter 1 presents the concept of truth algebra which is the basis of multi-valued interpretations of logics. Then, the concept of a logical structure suitable for defining a multi-valued first order logic to reason about systems is introduced. Multi-valued temporal logic mv-$CTL^*$ interpreted over multi-valued Kripke structures and multi-valued temporal logic of knowledge mv-$KCTL^*P$ interpreted over multi-agent multi-valued Kripke structures are recalled. The chapter ends with the presentation of the multi-valued model checking techniques found in the literature.

Chapter 2 introduces *multi-valued abstractions*. The abstractions are obtained by applying equivalence relations and then, the predicate symbols of the logic are re-defined to work properly on equivalence classes. As an equivalence class may contain more than one element and each element leads to a truth value for each predicate, to redefine a predicate on an equivalence class comes down to define a policy of recombination of truth values from some given set. Such a policy is called an *interpretation policy*. Abstrac-

tions of logical structures are then defined as pairs formed by an equivalence relation and an interpretation policy used to redefine the predicates, and give several preservation results for first order logic formulas. The abstractions of multi-valued Kripke structures are triples formed of an equivalence relation and two interpretation policies, one used to redefine the transition predicate and one used to redefine the atomic propositions. We prove their utility by the preservation results that follow. Finally, the abstractions of multi-agent multi-valued Kripke structures contain three interpretation policies, one being used to redefine the similarity relations. The preservation results we give involve formulas from the temporal logic of knowledge under Kleene's three-valued interpretation. Before giving abstractions of multi-valued Kripke structures, we provide a case study of using abstraction in the context of protection systems which model access control policies [63]. We propose two notions of simulation between protection systems and define a class of access control models that are simulated by access control models with a finite number of objects. Then, we show that several classes of protection systems from the literature fall into this class, notably the take-grant systems [78] and the monotonic typed access matrix systems with an acyclic creation graph [106]. By this we also unify and clarify the proof of decidability of the safety problem for these classes of protection systems.

*Equationally specified abstractions* are introduced and analyzed in Chapter 3. We define abstractions of data types modeled by membership algebras [91] and we show how to use abstractions specified by equations in the case of abstract data types. The membership algebras are a suitable logical framework in which a very wide range of total and partial equational specification formalisms can be naturally represented [91]. The membership algebra formalism is quite general and expressive, supports sub-sorts and overloading, and deals very well with errors and partiality. Moreover, membership algebra specifications can be efficiently implemented in systems like Maude [30]. The preservation results we obtain involve first order logic formulas and they are translated from the abstractions of logical structures. Moreover, the abstraction technique we propose generalizes and clarifies the nature of many abstraction techniques found in the literature, such as the technique of duplicating predicate symbols [26, 35, 6], shape analysis [96, 104], predicate abstraction [57, 36, 113], McMillan's approach [86] etc. For example, it is shown that the technique of duplicating predicate symbols, which is based on associating two versions to each formula, one used for validation and the other one used for refutation, consists of two abstractions based on the same congruence: one of them is weakly preserving (used for validation), and the other one is error preserving (used for refutation).

The approach in [92] which is also based on specifying abstractions by

equations, models systems by rewrite theories and the logic used to define the properties to be checked is $LTL$ under a 2-valued interpretation. It can be seen that this approach is a special case of the techniques described in this thesis.

Chapter 4 continues to apply this abstraction technique to dynamic data types. There have been proposed several approaches for modeling dynamic systems by universal algebras (see [2] for a survey on this topic). All the approaches are based on predicates which are added somehow to the signature, but they work outside the algebra. In the approach we propose we also add predicates to membership algebras, but they work inside the algebra (including the transition predicate too). This makes the formalism algebra-logic work unitarily. Now, the preservation results involve temporal logic formulas and they are translated from the abstractions of multi-valued Kripke structures introduced previously. The abstractions specified by equations are used again to define abstractions of abstract dynamic data types.

The main problem that arises when using abstraction techniques is to find the suitable abstraction or to refine an already existing abstraction in order to obtain a better one [76, 105, 66, 25, 87, 3, 80]. In Chapter 5, we prove that the types of abstraction of data types for Kleene's three-valued interpretation from [111] can be used in a refinement procedure. Moreover, we prove that the counterexample guided abstraction refinement procedure [25] works better when used with equationally specified abstractions.

# Chapter 1

# Multi-valued Logics

Multi-valued logics[1] provide an alternative to classical boolean logic for modeling and reasoning about systems. By adding new truth values, uncertainty and disagreement can be modeled explicitly and a variety of applications were found in databases [53], knowledge representation [55], machine learning [93], software and hardware verification [9, 10, 65].

A number of specific multi-valued logics have been proposed and studied. For example, Kleene [71] introduced a three-valued logic for reasoning with missing information, while Belnap [5] proposed a four-valued logic to handle inconsistent assertions in database systems. The fuzzy logic, where the truth values are all reals between 0 and 1, captures even more degrees of certainty. Each of these logics can be generalized to allow for different levels of uncertainty or disagreement. In practice, it is useful to be able to choose different multi-valued logics for different modeling tasks.

Many applications of multi-valued logics have been found in hardware and software verification. In the case of hardware verification, multi-valued logics can be applied for:

- building simulation tools and implementations of genuinely multi-valued circuits;
- modeling dynamic hazards by introducing pseudo states to find overlapping regions of competing signals (race detection) [13];
- test pattern generation by propagation of undefined or error values [21];
- verifying the implementation of gates on the basis of switch level models [64].

[1] In the literature on logic systems the terms multiple-valued, many-valued and multi-valued logic are used interchangeably.

Multi-valued logics play a crucial role to modeling and analyzing software systems. We need uncertainty because we may not know whether some behaviors should be possible, we need disagreement because we may have different stakeholders that disagree about how the systems should behave and we need to represent relative importance because some behaviors are essential and others may or may not be implemented. Moreover, to make model-checking practical for verification of real software systems, abstract models of the software behavior must be constructed. When working with abstractions, it is natural to consider three-valued logics, with the third value used to indicate elided information in the model [9], or to indicate the result of checking when a definite answer is not possible using the chosen abstraction [104, 18].

The remainder of this chapter provides a formal description of multi-valued logics. We begin, in Section 1.1, with the presentation of the structures we use for the set of truth values under which we will consider the multi-valued logics. Then, we describe in Section 1.2 the multi-valued first order logic and the multi-valued temporal logic with or without knowledge. The last section will survey the most important multi-valued model checking techniques.

## 1.1 Truth algebras

The set of truth values is any complete lattice together with a negation operator and some properties for it. Recall first a few basic concepts.

A *partial order* on a set $B$ is a binary relation on $B$ such that:

- (reflexivity) for any $a \in B$, $a \leq a$;
- (anti-symmetry) for any $a, b \in B$, $a \leq b$ and $b \leq a$ implies $a = b$;
- (transitivity) for any $a, b, c \in B$, $a \leq b$ and $b \leq c$ implies $a \leq c$.

We will denote by $<$ the relation $\leq - \{(a, a) | a \in B\}$ and we will say that $a \in B$ *immediately precede* $b \in B$, denoted $a \prec b$, if $a < b$ and there is no $c \in B$ such that $a < c < b$. By $\geq$ ($>$, $\succ$) we denote the inverse of $\leq$ ($<$, $\prec$). Also, $\downarrow x = \{y \in B \mid y \leq x\}$ and $\uparrow x = \{y \in B \mid y \geq x\}$.

From a partially ordered set we single out elements having special properties.

**Definition 1.1** Let $(B, \leq)$ be a partial order and $A \subseteq B$.

1. An element $b \in B$ is called a *lower bound* of $A$ if $b \leq x$, for any $x \in A$.

2. An element $b \in B$ is called a *greatest lower bound* (glb, for short) of $A$ if it is a lower bound and for any other lower bound $b'$ of $A$, we have $b' \leq b$.

3. An element $b \in B$ is called an *upper bound* of $A$ if $x \leq b$, for any $x \in A$.

4. An element $b \in B$ is called a *least upper bound* (lub, for short) of $A$ if it is an upper bound and for any other upper bound $b'$ of $A$, we have $b \leq b'$.

5. An element $a \in A$ is called a *least element* of $A$ if $a \leq x$, for any $x \in A$.

6. An element $a \in A$ is called a *greatest element* of $A$ if $x \leq a$, for any $x \in A$.

Clearly, if there exists a least upper bound (greatest lower bound) for a subset $A \subseteq B$ then it is unique.

**Definition 1.2** A partially ordered set $(B, \leq)$ is called a *lattice* if every finite subset of $B$ has a greatest lower bound and least upper bound. $(B, \leq)$ is called a *finite lattice* if the set $B$ is finite.

As usual, the least upper bound of a subset $A \subseteq B$ is denoted by $\vee A$, and the greatest lower bound of $A$ is denoted by $\wedge A$. When lattices will be used as domains for truth values, $\vee A$ ($\wedge A$) plays the role of the disjunction (conjunction) of all elements in $A$. 0 and 1 denote the least and, respectively, the greatest element of the lattice, if they exist.

**Definition 1.3** A lattice $(B, \leq)$ is called *complete* if every subset of $B$ has a greatest lower bound and least upper bound.

We can easily remark that any finite lattice is also complete and any complete lattice has a least and a greatest element.

Sometimes, for a more accurate modeling of truth values, distributivity is to be employed.

**Definition 1.4** A lattice $(B, \leq)$ is *distributive* if

- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and
- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$,

for any $a, b, c \in B$.

An element $x \in B$ of a lattice $(B, \leq)$ is called *join-irreducible* if $x = y$ or $x = z$ whenever $x$ can be written in the form $x = y \vee z$, for some $y, z \in B$. The least element, if it exists, is join-irreducible. Birkhoff's representation theorem [37] states that every element $b$ of a finite distributive lattice can be represented as the least upper bound of all join-irreducible elements less than or equal to $b$ in the lattice.

Now, we present *truth algebras*, structures over which we will define interpretations of multi-valued logics.

**Definition 1.5** A *truth algebra* [2] is a tuple $\mathcal{B} = (B, \wedge, \vee, \neg)$, where:

1. $(B, \leq)$ is a complete lattice, where $\leq$ is the binary relation on $B$ given by $a \leq b$ iff $a \vee b = b$, for any $a, b \in B$;

2. $\wedge$ and $\vee$ are the greatest lower bound and least upper bound operators, respectively;

3. $\neg : B \to B$ is a bijection such that $\neg 0 = 1$ and $\neg 1 = 0$.

$(B, \leq)$ is called the *lattice support* of $\mathcal{B}$. If the lattice support is finite or distributive, the truth algebra $\mathcal{B}$ will be called *finite* or *distributive*, respectively.

Notice that the completeness of the lattice $(B, \leq)$ ensures that $\wedge$ and $\vee$ always exist and they are unique.

**Example 1.1** A few examples of truth algebras are in order:

- classical 2-valued logics are based on the truth algebra whose lattice support has only 2 elements (Figure 1.1(a)). Since the lattice has only two elements, by the definition of a truth algebra, $\neg 0 = 1$ and $\neg 1 = 0$;

- Kleene's strong 3-valued logic [71] is based on the lattice in Figure 1.1(b) (the truth value $\perp$ means "undefined"). In this case, the negation operator is defined by $\neg \perp = \perp$ ($\neg 0 = 1$ and $\neg 1 = 0$ are implied by the definition of a truth algebra);

- the lattice in Figure 1.1(c) is the support for Belnap's 4-valued logic [5] which has been introduced for reasoning about inconsistent databases. The meaning of the truth values is as follows: 0 signifies that the property is false only, 1 signifies that the property is true only, $B$ signifies that the property is both true and false, and $N$ signifies that the

[2] We have coined the terminology of a "truth algebra" as an algebraic counterpart of a "c-complete lattice" used in [73].

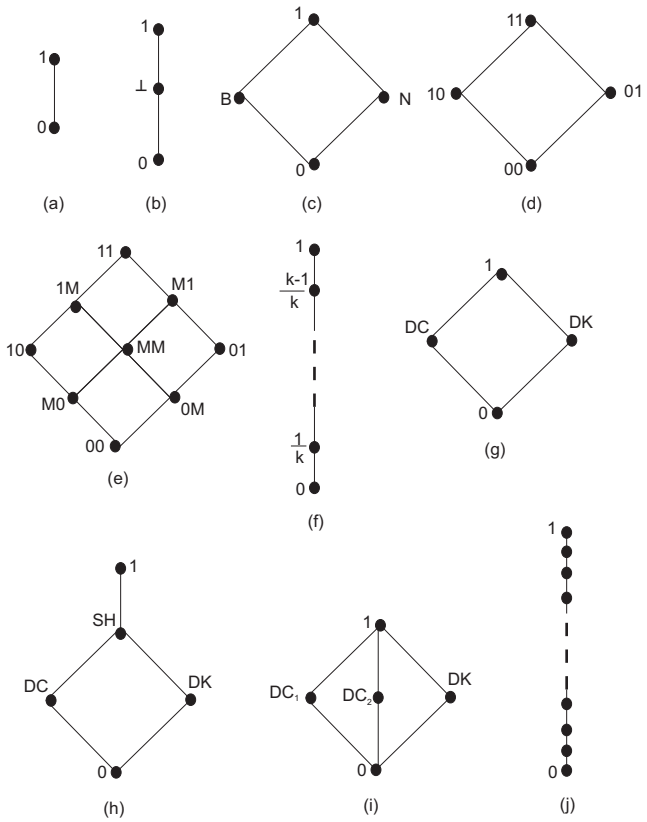


Figure 1.1: Examples of lattices of truth values

property is neither true nor false. The negation operator is defined by $\neg N = N$ and $\neg B = B$;

- the lattice in Figure 1.1(d) shows a logic that can be used for reasoning about disagreement between two knowledge sources [39]. The truth values have two components that specify the truth value of some formula from the point of view of each source. For example, 01 specifies the fact that the first knowledge source thinks that the formula is false and the second one thinks that it is true. The negation operator defines $\neg 10 = 01$ and $\neg 01 = 10$;

- the lattice in Figure 1.1(e) shows a nine-valued logic that can be used for reasoning about the disagreement between two sources, but also shows missing information in each source ($M$ is used to model missing information);

- the linear order in Figure 1.1(f) can be used to model different levels

of uncertainty: properties are more certain as they are interpreted to a value more close to 1;

- the lattice from Figure 1.1(g) has the same structure as Belnap's logic but this one is used to model different levels of uncertainty: besides the usual meaning for 0 and 1, $DC$ is used for values controlled by the environment and $DK$ for values controlled by the system, but not yet decided. The negation is defined by $\neg DC = DC$ and $\neg DK = DK$.

- the lattices from Figure 1.1(h) and Figure 1.1(i) add new values to the one from Figure 1.1(g). The first one adds a new value "should" (denoted $SH$) to express properties that are desired but not required (1 expresses properties that are required), and the second one replaces $DC$ by $DC_1$ and $DC_2$ because we suppose there exist two possible environments;

- the infinite lattice from Figure 1.1(j) models a set of truth values used in the fuzzy logic [81] (the reals between 0 and 1).

A particular case of truth algebras are *quasi-boolean algebras* [102, 7, 38, 17].

**Definition 1.6** A *quasi-boolean algebra* is a truth algebra $\mathcal{B} = (B, \wedge, \vee, \neg)$, where:

- its support $(B, \leq)$ is a finite distributive lattice;

- (De Morgan) $\neg(a \wedge b) = \neg a \vee \neg b$ and $\neg(a \vee b) = \neg a \wedge \neg b$, for all $a, b \in B$;

- (involution) $\neg\neg a = a$, for all $a \in B$;

- (antimonotonic) $a \leq b \Leftrightarrow \neg a \geq \neg b$, for all $a, b \in B$.

In the definition of a quasi-boolean algebra we may not require "$\neg 0 = 1$" and "$\neg 1 = 0$" because both of them can be easily obtained by using De Morgan laws. First, we remark that $a \wedge 0 = 0$, for any $a \in B$. Then, using the De Morgan laws, we obtain:

$$\neg 0 = \neg(a \wedge 0) = \neg a \vee \neg 0,$$

for any $a \in B$. Now, let $a_1$ be an element of $B$ such that $\neg a_1 = 1$ (this element exists because $\neg$ is a bijection). Then,

$$\neg 0 = 1 \vee \neg 0 = 1$$

which completes our proof. Analogously, we can prove that $\neg 1 = 0$.

**Example 1.2** All truth algebras in Figure 1.1, except for (h) and (i), are quasi-boolean. The truth algebra in Figure 1.1(h) is not a quasi-boolean algebra because there is no negation for $SH$; the truth algebra in Figure 1.1(i) is not quasi-boolean because its support is not a distributive lattice.

In [17] it was remarked that finite distributive lattices are symmetric about their horizontal axes and this symmetry is a suitable negation operator to form a quasi-boolean algebra.

**Definition 1.7** A lattice $(B, \leq)$ is *symmetric* if there exists a bijective function $H$ such that

- $a \leq b \Leftrightarrow H(a) \geq H(b)$, and
- $H(H(a)) = a$,

for any $a, b \in B$.

In some cases, truth algebras enjoy more properties such as non-contradiction and excluded middle.

**Definition 1.8** A *boolean algebra* is a tuple $\mathcal{B} = (B, \wedge, \vee, \neg, 0', 1')$ such that:

- $(B, \leq)$ is a distributive lattice, where $\leq$ is the binary relation on $B$ given by $a \leq b$ iff $a \vee b = b$, for all $a, b \in B$;
- $\wedge$ and $\vee$ are the greatest lower bound and least upper bound operators, respectively, and $\neg : B \to B$ is a bijection;
- $0'$ and $1'$ are two distinguished elements such that:
  - $a \vee 0' = a$ and $a \wedge 1' = a$, for any $a \in B$;
  - (non-contradiction) $a \wedge \neg a = 0'$, for any $a \in B$;
  - (excluded middle) $a \vee \neg a = 1'$, for any $a \in B$.

A boolean algebra is called *finite* if $B$ is finite.

We can remark that in the case of a boolean algebra $\mathcal{B} = (B, \wedge, \vee, \neg, 0', 1')$ whose support $(B, \leq)$ is complete, $0' = 0$ and $1' = 1$. Hence, finite boolean algebras are particular cases of quasi-boolean algebras.

**Example 1.3** All the truth algebras in Figure 1.1 except for (a) and (d) are not boolean algebras. Remember that the algebras in Figure 1.1(h) and Figure 1.1(i) were not even quasi-boolean while the others do not satisfy non-contradiction. For example, in the lattice from Figure 1.1(b) we have $\bot \wedge \neg\bot \neq 0$.

## 1.2 Logic systems

### 1.2.1 First-order logic

We introduce the concept of a *logical structure*, suitable both for modeling systems by membership algebras and for defining a logic for reasoning about systems.

Let $K$ be a non-empty set whose elements will be called *kinds* . A *$K$-kinded set* is a $K$-indexed family of sets $S = (S_k | k \in K)$. For a $K$-kinded set $S = (S_k | k \in K)$ and $w = k_1 \dots k_m \in K^+$, $m \geq 1$, we denote by $S_w$ the set $S_w = S_{k_1} \times \cdots \times S_{k_m}$ ($K^+$ stands for $K^* - \{\lambda\}$, where $K^*$ is the free monoid generated by $K$ and $\lambda$ is the unity of $K^*$).

A *$K$-kinded logical signature* is a pair $(\mathcal{B}, \Sigma_L)$, where $\mathcal{B}$ is a truth algebra and $\Sigma_L$ is a $K^+$-indexed set of pairwise disjoint sets

$$\Sigma_L = (\Sigma_{L,w} | w \in K^+).$$

The elements $k \in K^+$ are called *logical types* over $K$ and the elements $p \in \Sigma_{L,w}$ are called *predicate* or *logical symbols* of type $w$.

**Definition 1.9** Let $(\mathcal{B}, \Sigma_L)$ be a $K$-kinded logical signature. A *$(\mathcal{B}, \Sigma_L)$-logical structure* is a tuple $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$, where $S$ is a $K$-kinded set and $\Sigma_L^{\mathcal{S}}$ is a $K^+$-indexed set of predicate interpretations

$$\Sigma_L^{\mathcal{S}} = (\Sigma_{L,w}^{\mathcal{S}} | w \in K^+),$$

where $\Sigma_{L,w}^{\mathcal{S}} = \{p^{\mathcal{S}} : S_w \to B | p \in \Sigma_{L,w}\}$ is a $\Sigma_{L,w}$-indexed set of functions from $S_w$ into $B$.

Given $(\mathcal{B}, \Sigma_L)$ a $K$-kinded logical signature and an at most countable $K$-kinded set of variables $X$, the set of *first order formulas* over $(\mathcal{B}, \Sigma_L)$ and $X$ is defined as follows:

1. *atomic formulas:*
   (a) $p(x_1, \dots, x_m)$ is an atomic formula, for any predicate symbol $p$ of logical type $w = k_1 \dots k_m \in K^+$ and $x_i \in X_{k_i}$, $1 \leq i \leq m$;
2. *formulas:*
   (a) every atomic formula is a formula;
   (b) if $\phi_1$ and $\phi_2$ are formulas then $\neg\phi_1$, $(\phi_1 \vee \phi_2)$, and $(\phi_1 \wedge \phi_2)$ are formulas;

(c) if $x$ is a variable and $\phi$ is a formula, then $((\exists x)\phi)$ and $((\forall x)\phi)$ are formulas.

Denote by $\mathcal{L}^O(\Sigma_L, X)$, where $O \subseteq \{\wedge, \vee, \neg\}$, the set of first order formulas over $(\mathcal{B}, \Sigma_L)$ and $X$ that use only operators from the set $O$, the quantifier $\forall$ if $\wedge \in O$, and the quantifier $\exists$ if $\vee \in O$ (when $O = \{\wedge, \vee, \neg\}$, we omit the superscript "$O$" from the notation $\mathcal{L}^O(\Sigma_L, X)$).

An assignment of $X$ into $\mathcal{S}$ is a function $\gamma : X \to S$ such that $\gamma(x) \in S_k$, for any $x \in X_k$. $\Gamma(X, \mathcal{S})$ stands for the set of all assignments of $X$ into $\mathcal{S}$.

Given a $(\mathcal{B}, \Sigma_L)$-logical structure $\mathcal{S}$ each first order formula $\phi$ induces a function $\mathcal{I}_{\mathcal{S}}(\phi)$ from the set $\Gamma(X, \mathcal{S})$ into $B$, as follows:

- $\mathcal{I}_{\mathcal{S}}(p(x_1, \ldots, x_m))(\gamma) = p^{\mathcal{S}}(\gamma(x_1), \ldots, \gamma(x_m))$, for any $p$ of logical type $w = k_1 \ldots k_m \in K^+$ and $x_i \in X_{k_i}$, $1 \leq i \leq m$;
- $\mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2) = \mathcal{I}_{\mathcal{S}}(\phi_1) \wedge \mathcal{I}_{\mathcal{S}}(\phi_2)$, for any formulas $\phi_1$ and $\phi_2$;
- $\mathcal{I}_{\mathcal{S}}(\phi_1 \vee \phi_2) = \mathcal{I}_{\mathcal{S}}(\phi_1) \vee \mathcal{I}_{\mathcal{S}}(\phi_2)$, for any formulas $\phi_1$ and $\phi_2$;
- $\mathcal{I}_{\mathcal{S}}(\neg\phi_1) = \neg\mathcal{I}_{\mathcal{S}}(\phi_1)$, for any formula $\phi$;
- $\mathcal{I}_{\mathcal{S}}((\exists x)\phi)(\gamma) = \bigvee_{a \in S_k} \mathcal{I}_{\mathcal{S}}(\phi)(\gamma[x/a])$, for any formula $\phi$ and any $x \in X_k$;
- $\mathcal{I}_{\mathcal{S}}((\forall x)\phi)(\gamma) = \bigwedge_{a \in S_k} \mathcal{I}_{\mathcal{S}}(\phi)(\gamma[x/a])$, for any formula $\phi$ and any $x \in X_k$.

We emphasize that "$\bigvee_{a \in S_k}$" and "$\bigwedge_{a \in S_k}$" in the definition above are the least upper bound and the greatest lower bound of some set of truth values.

$\mathcal{I}_{\mathcal{S}}(\phi)$ is called the *interpretation function* of $\phi$ into $\mathcal{S}$. If $\phi$ is a formula, we say that $\phi$ *has the truth value* $b \in B$ *in* $\mathcal{S}$, and denote this by $[\phi]^{\mathcal{S}} = b$, if $\mathcal{I}_{\mathcal{S}}(\phi)(\gamma) = b$, for all $\gamma \in \Gamma(X, \mathcal{S})$.

### 1.2.2 Temporal logic

The temporal logic $CTL^*$ [27] describes sequences of transitions between states in a reactive system which interacts with and continuously responds to its environment. This logic uses atomic propositions and boolean operators to build up formulas describing properties of states. Moreover, path operators and quantifiers are introduced to describe transitions between states.

The temporal logic $CTL^*$ is defined over some set $AP$ of atomic propositions and it contains two types of formulas, *state* and *path* formulas . Their syntax is given by the following rules ($p \in AP$, $\varphi$ is a state formula, and $\psi$ is a path formula):

- $true$, $false$ and $p$ are state formulas, for any $p \in AP$;
- if $\varphi_1$ and $\varphi_2$ are state formulas, then so are $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, and $\varphi_1 \wedge \varphi_2$;
- if $\psi$ is a path formula, then $\forall\psi$ and $\exists\psi$ are state formulas;
- each state formula is a path formula;
- if $\psi_1$ and $\psi_2$ are path formulas, then so are $\neg\psi_1$, $\psi_1 \vee \psi_2$, $\psi_1 \wedge \psi_2$, $\mathbf{X}\psi_1$, $\overline{\mathbf{X}}\psi_1$, $\psi_1\,\mathbf{U}\psi_2$, and $\psi_1\,\mathbf{R}\psi_2$.

We abbreviate $\mathbf{F}\psi = true\ \mathbf{U}\ \psi$ and $\mathbf{G}\psi = false\ \mathbf{R}\ \psi$. The meaning of the operators above are as follows: $\mathbf{X}$ and $\overline{\mathbf{X}}$ are the strong and weak versions of "next-time", $\mathbf{U}$ is "until", $\mathbf{R}$ is "releases", $\mathbf{F}$ is "eventually", $\mathbf{G}$ is "always", $\forall$ is "for all paths", and $\exists$ is "there exists a path".

$CTL^*$ is the set of all state formulas generated by the rules presented above; $CTL^*_+$ is the subset of $CTL^*$ consisting of formulas without negation, $CTL$ is the subset of $CTL^*$ consisting of formulas in which each future time operator is immediately preceded by a path quantifier, $LTL$ is the subset of $CTL^*$ consisting of formulas of the form $\forall\psi$, where $\psi$ is a path formula in which the only state subformulas permitted are atomic propositions, and $\forall CTL^*$ ($\exists CTL^*$) is the subset of $CTL^*$ consisting of formulas that do not contain $\exists$ ($\forall$). In order to avoid implicit existential (universal) paths quantifiers resulting from the use of negation in $\forall CTL^*$ ($\exists CTL^*$) formulas, we assume that the path quantifiers are not in the scope of a negation.

**Example 1.4** A few examples of $CTL^*$ formulas together with their intuitive meaning are in order:

- $\exists\,\mathbf{G}\,\mathbf{F}\,p$: there exists a path that contains infinitely many states in which $p$ is true;
- $\exists\ (\mathbf{G}\ (p \Rightarrow \mathbf{X}\ p))$: there exists a path along which $p$ holds continuously from the first state in which it holds;
- $\forall\ (p \Rightarrow \mathbf{F}\ q)$: if $p$ holds then, by any possible path, we can reach a state in which $q$ holds;
- $\forall\ \mathbf{F}\,\mathbf{G}\,p$: along every path, there exists some state from which $p$ will hold forever;
- $\forall\,\mathbf{G}\ (p \Rightarrow \forall\,\mathbf{F}\ q)$: if we reach a state satisfying $p$, then we will eventually reach a state satisfying $q$;

- $\forall$ **G** ($\exists$ **F** $p$): from any state it is possible to get to a state in which $p$ holds.

The third and the fourth formulas in Example 1.4 are $LTL$ formulas while the fifth and the sixth are $CTL$ formulas. The fourth one has the property that there is no equivalent $CTL$ formula and for the sixth one there is no equivalent $LTL$ formula.

"Multi-valued semantics" of $CTL^*$ are given by means of multi-valued Kripke structures [51, 52].

**Definition 1.10** A *multi-valued Kripke structure* (*mv-Kripke structure*) over a set of atomic propositions $AP$ and a truth algebra $\mathcal{B} = (B, \wedge, \vee, \neg)$ is a tuple $M = (Q, R, L)$, where:

- $Q$ is the set of states;
- $R : Q \times Q \to B$ is the (multi-valued) transition predicate;
- $L : Q \to (AP \to B)$ is a function which associates to any state $q$ the "truth values" of the atomic propositions in state $q$.

Let $M$ be an mv-Kripke structure over $AP$ and $\mathcal{B}$, and $D$ be a subset of $B$. An *infinite $D$-sequence* of states is a sequence $\pi = q_0 q_1 \ldots$ such that $R(q_i, q_{i+1}) \in D$, for all $i \geq 0$. The length of $\pi$ is $\infty$. A *maximal finite $D$-sequence* of states is a sequence $\pi = q_0 q_1 \ldots q_n$ such that $R(q_i, q_{i+1}) \in D$, for all $0 \leq i < n$, and $R(q_n, q) \notin D$, for all $q \in Q$. The length of $\pi$ is $n + 1$. A *$D$-path* of $M$ is either an infinite $D$-sequence or a maximal finite $D$-sequence of states. $\mathsf{Paths}(M, D, q)$ stands for the set of all $D$-paths of $M$ starting in state $q$, $\mathsf{Paths}(M, D)$ stands for the set of all $D$-paths of $M$, $\pi(i)$ stands for the state $q_i$, and $\pi^i$ is the suffix of $\pi$ starting at $q_i$, for any $0 \leq i < |\pi|$.

The *truth value of a $CTL^*$ state formula $\phi$ in a state $q$ of $M$* (denoted $[\phi]_q^M$) and the *truth value of a $CTL^*$ path formula $\psi$ along a $D$-path $\pi$ in $M$* (denoted $[\psi]_\pi^M$) are defined as follows ($p \in AP$, $\varphi$, $\varphi_1$, and $\varphi_2$ are state formulas, and $\psi$, $\psi_1$, and $\psi_2$ are path formulas):

$$
\begin{array}{lcl}
[true]_q^M & = & 1; \\
[false]_q^M & = & 0; \\
[p]_q^M & = & L(q)(p); \\
[\neg\varphi]_q^M & = & \neg[\varphi]_q^M; \\
[\varphi_1 \wedge \varphi_2]_q^M & = & [\varphi_1]_q^M \wedge [\varphi_2]_q^M; \\
[\varphi_1 \vee \varphi_2]_q^M & = & [\varphi_1]_q^M \vee [\varphi_2]_q^M;
\end{array}
$$

$$
\begin{array}{lcl}
[\varphi]^M_\pi & = & [\varphi]^M_{\pi(0)}; \\
[\neg\psi]^M_\pi & = & \neg[\psi]^M_\pi; \\
[\psi_1 \wedge \psi_2]^M_\pi & = & [\psi_1]^M_\pi \wedge [\psi_2]^M_\pi; \\
[\psi_1 \vee \psi_2]^M_\pi & = & [\psi_1]^M_\pi \vee [\psi_2]^M_\pi; \\
[\mathbf{X}\psi]^M_\pi & = & (|\pi| > 1) \wedge R(\pi(0), \pi(1)) \wedge [\psi]^M_{\pi 1}; \\
[\overline{\mathbf{X}}\psi]^M_\pi & = & (|\pi| \leq 1) \vee (R(\pi(0), \pi(1)) \wedge [\psi]^M_{\pi 1}); \\
[\psi_1 \mathbf{U} \psi_2]^M_\pi & = & [\psi_2]^M_{\pi^0} \vee \bigvee_{0<i<|\pi|}([\psi_2]^M_{\pi^i} \wedge \bigwedge_{0\leq j<i}([\psi_1]^M_{\pi^j} \wedge R(\pi(j), \pi(j+1)))); \\
[\psi_1 \mathbf{R} \psi_2]^M_\pi & = & [\psi_2]^M_{\pi^0} \wedge \bigwedge_{0<i<|\pi|}((R(\pi(i-1), \pi(i)) \wedge [\psi_2]^M_{\pi^i}) \vee \bigvee_{0\leq j<i}([\psi_1]^M_{\pi^j})); \\
[\forall\psi]^M_q & = & \bigwedge_{\pi\in \mathsf{Paths}(M,D,q)}[\psi]^M_\pi; \\
[\exists\psi]^M_q & = & \bigvee_{\pi\in \mathsf{Paths}(M,D,q)}[\psi]^M_\pi;
\end{array}
$$

(it is implicitly assumed that the values of the standard predicates $<$, $\leq$, $>$, and $\geq$ are 0 and 1, the least and greatest element of $\mathcal{B}$).

**Remark 1.1** Throughout this thesis many results will involve the positive version $CTL^*_+$ of $CTL^*$. However, if this logic is considered on quasi-boolean algebras, this is not a restriction. On the basis of De Morgan laws, we can suppose without loss of generality that any formula in $CTL^*$ is in the *negation normal form*, i.e. the negation is only applied to the atomic propositions and then, we can use a copy of the set of atomic propositions for their negations to obtain an equivalent formula in $CTL^*_+$.

When the logic $CTL^*$ is interpreted over mv-Kripke structures it is usually called *multi-valued* $CTL^*$ (*mv-*$CTL^*$).

### 1.2.3 Temporal logic of knowledge

Temporal logic of knowledge $KCTL^*P$ [49] is suitable for reasoning about knowledge in the context of *multi-agent systems*, i.e. systems consisting in a collection of interacting agents. The modeling of knowledge is based on "possible worlds". The intuition is that if an agent does not have complete knowledge about the world, he will consider a number of possible worlds (these are given by a *similarity relation*). The agent is said to *know a fact* $\phi$ if $\phi$ holds at all agent's possible worlds.

Let $AP$ be a set of atomic propositions and $n$ the number of agents in the system. As in the case of $CTL^*$, there are two types of formulas, *state* and *path* formulas . The rules describing its syntax include the ones of $CTL^*$ together with rules for past time operators and knowledge operators:

- if $\psi_1$ and $\psi_2$ are path formulas, then so are $\mathbf{Pr}\ \psi_1$, $\psi_1\,\mathbf{S}\,\psi_2$ and $\psi_1\,\mathbf{B}\,\psi_2$;

- if $\varphi$ is a state formula, then so are $\mathbf{K}_i\,\varphi$ and $\mathbf{P}_i\,\varphi$, for any $1 \leq i \leq n$.

We abbreviate $\mathbf{A}\psi = \psi\ \mathbf{B}\ false$ and $\mathbf{O}\psi = true\ \mathbf{S}\ \psi$. The past time path operators have the following meaning [82]: **Pr** is "previous", **A** is "always in the past", **O** is "once", **S** is "since", and **B** is "back to". The knowledge operators have the meaning [49]: $\mathbf{K}_i$ is "agent $i$ knows $\phi$" and $\mathbf{P}_i$ is "agent $i$ thinks that $\phi$ is possible".

$KCTL^*P$ is the set of all state formulas generated by the rules presented above. We denote by $\forall KCTL^*P$ ($\exists KCTL^*P$) the subset of $KCTL^*P$ consisting of formulas that do not contain $\exists$ and $\mathbf{P}_i$ ($\forall$ and $\mathbf{K}_i$). In order to avoid implicit existential (universal) paths quantifiers and knowledge operators $\mathbf{P}_i$ ($\mathbf{K}_i$) resulting from the use of negation in $\forall KCTL^*P$ ($\exists KCTL^*P$) formulas, we assume that the quantifiers and the knowledge operators are not in the scope of a negation.

**Example 1.5** A few examples of $KCTL^*P$ formulas together with their intuitive meaning are in order:

- $\mathbf{P}_2\mathbf{K}_1 p$ : agent 2 considers possible the fact that agent 1 knows $p$;
- $\mathbf{K}_1(bit = 0) \vee \mathbf{K}_1(bit = 1)$: agent 1 knows the value of *bit*;
- $\mathbf{K}_1\exists\mathbf{F}p \vee \mathbf{K}_1\exists\mathbf{F}\neg p$ : agent 1 knows that either $p$ or $\neg p$ will hold eventually, from all the states he considers possible;
- $\mathbf{K}_2\mathbf{K}_1 p \wedge \neg\mathbf{K}_1\mathbf{K}_2\mathbf{K}_1 q$ : agent 2 knows that agent 1 knows $p$ but agent 1 does not know that agent 2 knows that agent 1 knows $q$;

"Multi-valued semantics" of $KCTL^*P$ are given by means of multi-agent multi-valued Kripke structures which extend the *mv modal epistemic models* [74] by the use of multi-valued similarity relations.

**Definition 1.11** Let $AP$ be a set of atomic propositions and $\mathcal{B} = (B, \wedge, \vee, \neg)$ a truth algebra. A *multi-agent multi-valued Kripke structure* (*multi-agent mv-Kripke structure*) over $AP$ and $\mathcal{B}$ is a tuple $M = (Q, R, L, (\sim_i |\ 1 \leq i \leq n))$, where $(Q, R, L)$ is an mv-Kripke structure over $AP$ and $\mathcal{B}$ and $\sim_i$: $Q \times Q \to B$, for all $1 \leq i \leq n$, is the similarity relation of agent $i$ . Each similarity relation $\sim_i$ satisfies the following properties:

- *reflexivity*: $\sim_i (x, x) = 1$, for any $x \in Q$;
- *symmetry*: $\sim_i (x, y) = \sim_i (y, x)$, for any $x, y \in Q$;
- *transitivity*: $\sim_i (x, z) = \sim_i (x, y) \wedge \sim_i (y, z)$, for any $x, y, z \in Q$.

As we can remark, $\sim_i$ is a multi-valued binary relation on $Q$. "$\sim_i (x, y) = b$" means that agent $i$ associates the truth value $b$ to the similarity of $x$ and $y$.

Given a multi-agent mv-Kripke structure $M$ over $AP$ and $\mathcal{B}$, and given a subset $D$ of $B$, define the set of all $D$-paths of $M$ starting in state $q$, denoted $\mathsf{Paths}(M, D, q)$, exactly as for mv-Kripke structures. Again, $\mathsf{Paths}(M, D)$ stands for the set of all $D$-paths of $M$. Moreover, we will call a *point* any pair $(\pi, m)$, where $\pi \in \mathsf{Paths}(M, D)$ and $m \in \mathbf{N}$ with $0 \leq m < |\pi|$. The set of points of the Kripke structure $M$ will be denoted by $\mathsf{Points}(M, D)$.

For any multi-agent mv-Kripke structure, we can obtain an interpreted system by unwinding the transition predicate. The labeling function on points is compatible with the one on the corresponding states.

**Definition 1.12** Let $M = (Q, R, L, (\sim_i |\ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure. The *interpreted system* corresponding to $M$ is a triple $I = (\mathsf{Paths}(M, D), L^I, (\sim_i^I |\ 1 \leq i \leq n))$, where

- $L^I : \mathsf{Points}(M, D) \to (AP \to B)$ is the interpretation function for the atomic propositions in $M$ defined by $L^I(\pi, m)(p) = L(\pi(m))(p)$, for any $\pi \in \mathsf{Paths}(M, D)$ and $0 \leq m < |\pi|$;
- $\sim_i^I$: $\mathsf{Points}(M, D) \times \mathsf{Points}(M, D) \to B$, for any $1 \leq i \leq n$, is the similarity relation defined by $\sim_i^I ((\pi, m), (\pi', m')) = \sim_i (\pi(m), \pi'(m'))$.

From now on we will make no distinction between $L$ and its extension to points $L^I$. The same holds for the similarity relations.

We now give the semantics of $KCTL^*P$, which for the fragment that includes only future time and knowledge is similar to the multi-valued semantics in [74]. If $\phi$ is a formula in $KCTL^*P$ and $I = (\mathsf{Paths}(M, D), L, (\sim_i |\ 1 \leq i \leq n))$ is an interpreted system we define the *truth value of $\phi$ in the point* $(\pi, m)$, denoted by $[\phi]^I_{(\pi,m)}$, in the following way ($p \in AP$, $\varphi$, $\varphi_1$, and $\varphi_2$ are state formulas, and $\psi$, $\psi_1$, and $\psi_2$ are path formulas):

$$
\begin{array}{lcl}
[true]^I_{(\pi,m)} & = & 1; \\
[false]^I_{(\pi,m)} & = & 0; \\
[p]^I_{(\pi,m)} & = & L(\pi, m)(p); \\
[\neg p]^I_{(\pi,m)} & = & \neg L(\pi, m)(p); \\
[\phi_1 \wedge \phi_2]^I_{(\pi,m)} & = & [\phi_1]^I_{(\pi,m)} \wedge [\phi_2]^I_{(\pi,m)}; \\
[\phi_1 \vee \phi_2]^I_{(\pi,m)} & = & [\phi_1]^I_{(\pi,m)} \vee [\phi_2]^I_{(\pi,m)}; \\
[\mathbf{X}\phi]^I_{(\pi,m)} & = & |\pi^m| > 1 \wedge R(\pi(m), \pi(m+1)) \wedge [\phi]^I_{(\pi,m+1)};
\end{array}
$$

$$
\begin{array}{lcl}
[\overline{\mathbf{X}}\phi]^I_{(\pi,m)} & = & |\pi^m| \leq 1 \vee (R(\pi(m),\pi(m+1)) \wedge [\phi]^I_{(\pi,m+1)}); \\
[\phi_1 \mathbf{U} \phi_2]^I_{(\pi,m)} & = & [\phi_2]^I_{(\pi,m)} \vee \bigvee_{0<i<|\pi^m|}([\phi_2]^I_{(\pi,m+i)} \wedge \\
 & & \bigwedge_{0\leq j<i}([\phi_1]^I_{(\pi,m+j)} \wedge R(\pi(m+j),\pi(m+j+1)))); \\
[\phi_1 \mathbf{R} \phi_2]^I_{(\pi,m)} & = & [\phi_2]^I_{(\pi,m)} \wedge \bigwedge_{0<i<|\pi^m|}((R(\pi(m+i-1),\pi(m+i)) \wedge [\psi]^I_{\pi^{m+i}}) \\
 & & \vee \bigvee_{0\leq j<i}([\phi]^I_{\pi^{m+j}})); \\
[\mathbf{Pr}\ \phi]^I_{(\pi,m)} & = & m = 0 \vee (R(\pi(m-1),\pi(m)) \wedge [\phi]^I_{(\pi,m-1)}); \\
[\phi_1 \mathbf{S} \phi_2]^I_{(\pi,m)} & = & [\phi_2]^I_{(\pi,m)} \vee \bigvee_{0\leq i<m}([\phi_2]^I_{(\pi,i)} \wedge \\
 & & \bigwedge_{i<j\leq m}([\phi_1]^I_{(\pi,j)} \wedge R(\pi(j-1),\pi(j)))); \\
[\phi_1 \mathbf{B} \phi_2]^I_{(\pi,m)} & = & [\phi_1]^I_{(\pi,m)} \wedge \bigwedge_{0\leq i<m}((R(\pi(i),\pi(i+1)) \wedge [\phi_1]^I_{\pi^i}) \\
 & & \vee \bigvee_{i<j\leq m}([\phi_2]^I_{\pi^j})); \\
[\exists \phi]^I_{(\pi,m)} & = & \bigvee_{\pi'[1..m]=\pi[1..m]} [\phi]^I_{(\pi',m)}; \\
[\forall \phi]^I_{(\pi,m)} & = & \bigwedge_{\pi'[1..m]=\pi[1..m]} [\phi]^I_{(\pi',m)}; \\
[\mathbf{K}_i \phi]^I_{(\pi,m)} & = & \bigwedge_{\sim_i((\pi,m),(\pi',m'))\neq 0} \big( \sim_i ((\pi,m),(\pi',m')) \wedge [\phi]^I_{(\pi',m')} \big); \\
[\mathbf{P}_i \phi]^I_{(\pi,m)} & = & \bigvee_{\sim_i((\pi,m),(\pi',m'))\neq 0} \big( \sim_i ((\pi,m),(\pi',m')) \wedge [\phi]^I_{(\pi',m')} \big);
\end{array}
$$

(it is implicitly assumed that the values of the standard predicates $<$, $\leq$, $>$, and $\geq$ are 0 and 1, the least and greatest element of $\mathcal{B}$).

When the logic $KCTL^*P$ is interpreted over multi-agent mv-Kripke structures, it is usually called *multi-valued* $KCTL^*P$ (*mv-*$KCTL^*P$, for short).

## 1.3 Multi-valued model checking

Model checking multi-valued logics have been proposed by many researchers [16, 72, 19, 17, 20, 60, 11, 73, 74]. Two main approaches have gained attention:

- the *reduction approach* [72, 60, 11, 73, 74] which transforms an instance of the multi-valued model checking problem into a set of instances of the 2-valued model-checking problem (in this way we can use previously known model checking algorithms);
- the *direct approach* [16, 19, 17, 20, 11] which introduces specialized algorithms for multi-valued model checking.

We can not say which of the above approaches is better. The complexity is always the product between the size of the lattice and the complexity of 2-valued model checking. As already mentioned in [11], the reduction

approach benefits from the fact that it can be implemented using already existing model checkers but the direct approach can work in a more "on the fly" manner. In the reduction approach, we have to construct a set of 2-valued Kripke structures starting only from the initial mv-Kripke structure and the truth algebra and then check the formula on all these models. In the direct approach we can take into consideration all three inputs: the mv-Kripke structure, the truth algebra and the formula in order to guide our check.

### 1.3.1 Reduction to 2-valued model checking

#### The approach using Birkhoff's representation theorem

Reduction techniques for multi-valued model checking to 2-valued model checking are introduced in [72, 60, 11, 74]. Although the authors of [60, 11] use $\mu$-calculus as their supporting logic, straightforward particularizations can be made for $CTL^*$. Moreover, the $\mu$-calculus combined with knowledge modalities used in [74] permits us to extract a reduction result even for $KCTL^*$ on multi-agent mv-Kripke structures with 2-valued similarity relations.

The idea of all the techniques above is the same. We define for each mv-Kripke structure $M$ over a quasi-boolean algebra $\mathcal{B}$, a set of 2-valued Kripke structures $\{M_x | x \in \mathcal{J}(B)\}$[3], where $\mathcal{J}(B)$ is the set of join-irreducible elements of $(B, \leq)$. Then, we use the result of the 2-valued model checking on each of these 2-valued Kripke structures to obtain the truth value of the formula in the initial structure.

We proceed with the formalization of the reduction result using the approach in [72, 74]. In [72], the authors give a preliminary reduction result and complete solutions only for three classes of quasi-boolean algebras. Later, in [74], they extend the method for the temporal logic of knowledge and offer a complete solution for any quasi-boolean algebra.

Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a quasi-boolean algebra. The result in [72] considers a function $f : B \to B$ with $f(B) \neq B$ which preserves arbitrary bounds, and proves that multi-valued model checking over $\mathcal{B}$ can be reduced to model checking over a quasi-boolean algebra induced by $f(B)$ (if $|f(B)| = 2$ then we deal with the standard 2-valued model checking for $CTL^*$).

**Theorem 1.1** [72] Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a quasi-boolean algebra and

[3]The use of $\mu$-calculus in [60, 11, 74] complicates the construction of the Kripke structures $M_x$ because in the semantics of $\mu$-calculus we also need the negation of the transition predicate.

$f : B \to B$ be a function with $f(B) \neq B$ and which preserves arbitrary bounds, i.e.

$$f(\wedge B') = \bigwedge_{b \in B'} f(b) \text{ and } f(\vee B') = \bigvee_{b \in B'} f(b),$$

for any $B' \subseteq B$. Further, let $M = (Q, R, L)$ be an mv-Kripke structure over $\mathcal{B}$ and some set of atomic propositions $AP$ and let $M' = (Q, R', L')$ be an mv-Kripke structure over a quasi-boolean algebra induced by $f(B)$ and $AP$ such that $R'(q, q') = f(R(q, q'))$ and $L'(q)(p) = f(L(q)(p))$, for any $q, q' \in Q$ and $p \in AP$.

Then, for any state (path) $CTL^*_+$ formula $\phi$ ($\psi$), it holds:

- $[\phi]^M_q \in f^{-1}(b)$ iff $[\phi]^{M'}_q = b$;
- $[\psi]^M_\pi \in f^{-1}(b)$ iff $[\psi]^{M'}_\pi = b$,

for any $q \in Q'$ and $\pi \in \mathsf{Paths}(M, B - \{0\})$.

In [74] this result is extended for multi-agent mv-Kripke structures with 2-valued similarity relations and formulas expressed in the $\mu$-calculus combined with knowledge modalities. This extension imposes another restriction on the function $f$ because in the semantics of $\mu$-calculus we also need the negation of the transition predicate, which can be discarded when considering formulas expressed in $KCTL^*_+$.

If $M = (Q, R, L)$ is an mv-Kripke structure as above, Theorem 1.1 permits us to answer questions of the form "$[\phi]^M_q \in f^{-1}(b)$?" or "$[\psi]^M_\pi \in f^{-1}(b)$?". In order to offer the exact truth value of some formula, the following procedure is proposed in [72]:

- define $k$ mappings $f_1$,..., $f_k$ such that:
  - $f_1(B)$,...,$f_k(B)$ are lattices for which we already have model checking algorithms;
  - for any $b \in B$ there exist a set of indexes $\{i_1, \ldots, i_p\} \subseteq \{1, \ldots, k\}$ and $b_{i_j} \in f_{i_j}(B)$, for all $1 \leq j \leq p$, such that:

$$\bigcap_{1 \leq j \leq p} f^{-1}_{i_j}(b_{i_j}) = \{b\}.$$

- if $M_1$,..., $M_k$ are the mv-Kripke structures defined as in Theorem 1.1 over $f_1(B)$,..., $f_k(B)$, respectively, then:

$$[\phi]^M_q = b \text{ iff } \left([\phi]^{M_{i_1}}_q = b_{i_1} \wedge \cdots \wedge [\phi]^{M_{i_p}}_q = b_{i_p}\right).$$

[72] provides solutions for finding mappings like the ones above only for linear orders, products of two linear orders, and $2 \times 2 + 2$. For instance, if $\mathcal{B}$ with $B = \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}, 1\}$ is a linear order, we define $k$ mappings $f_k : B \to \{0, 1\}$ in the following way:

- $f_i(x) = 0$ iff $x < \frac{i}{k}$ and
- $f_i(x) = 1$ iff $x \geq \frac{i}{k}$,

for any $1 \leq i < k$, and $f_k(x) = 1$ if $x = 1$ and $f_k(x) = 0$, otherwise.

Notice that all $f_i$, $1 \leq i \leq k$, preserve arbitrary bounds and also, for any $\frac{j}{k}$ with $1 \leq j \leq k-1$, we can find the set of indexes $\{j, j+1\}$, $b_j = 1$ and $b_{j+1} = 0$ such that:

$$\frac{j}{k} = f_j(1) \cap f_{j+1}(0).$$

Consequently, for any $CTL^*_+$ formula:

$$[\phi]^M_q = \frac{j}{k} \text{ iff } [\phi]^{M_j}_q = 1 \text{ and } [\phi]^{M_{j+1}}_q = 0.$$

For the truth values 0 and 1, we can easily obtain $[\phi]^M_q = 0$ iff $[\phi]^{M_1}_q = 0$ and $[\phi]^M_q = 1$ iff $[\phi]^{M_k}_q = 1$.

In [74], a general method is offered, to obtain mappings like the ones above for any quasi-boolean algebra $\mathcal{B}$. The technique is based on Birkhoff's representation theorem.

Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a quasi-boolean algebra like above. It is proved that for any irreducible element $x$ of $(B, \leq)$, the function $f_x : B \to \{0, 1\}$ defined by $f_x(b) = 1$, for any $b \geq x$ and $f_x(b) = 0$, otherwise, preserves arbitrary bounds.

Then, by Birkhoff's representation theorem for finite distributive lattices, which states that every element $b$ of such a lattice can be represented as the least upper bound of all the join-irreducible elements less than or equal to $b$ in the lattice, we obtain that for any $b \in B$, there exist the set $\mathcal{J}(B) \cap b\downarrow = \{i_1, \ldots, i_p\}$ such that $i_1\uparrow \cap \cdots \cap i_p\uparrow = \{b\}$. Consequently, for any $b \in B$, we can find the set of indexes $\{i_1, \ldots, i_p\}$ and $b_{i_1} = \cdots = b_{i_p} = 1$ such that

$$\bigcap_{1 \leq j \leq p} f^{-1}_{i_j}(1) = \{b\}.$$

Moreover, the set of indexes above is unique and consequently we can derive the following result.

**Theorem 1.2** [74] Let $M = (Q, R, L)$ be an mv-Kripke structure over a quasi-boolean algebra $\mathcal{B}$ and a set of atomic propositions $AP$. Then,

$$[\phi]_q^M = \bigvee \{x \in \mathcal{J}(B) | [\phi]_q^{M_x} = 1\},$$

for any $q \in Q$ and $\phi$ a $CTL_+^*$ formula.

**Complexity** Theorem 1.2 implies that the complexity of multi-valued model checking over a quasi-boolean lattice $\mathcal{B}$ is bounded by $nM$, where $n$ is the number of join-irreducible elements of $(B, \leq)$ and $M$ is the complexity of 2-valued model-checking.

### The approach using designated values

Another reduction technique from multi-valued model checking to 2-valued model checking is the one in [73]. In this case, we start with a weaker multi-valued model checking problem: instead of computing the exact truth value of some formula, we are interested in finding if the truth value of some formula is in some set of *designated values*.

As the authors of [73] claim, the designated values should be the counterpart of truth in classical logic: a multi-valued formula is considered to be valid if its truth value is designated.

We now proceed with the description of the reduction in [73]. If $\mathcal{B} = (B, \wedge, \vee, \neg)$ is a truth algebra, a set of designated values $D \subseteq B$ must have the following properties:

- $D$ is a proper non-empty subset of $B$;
- $D$ is *upward closed under* $\leq$, i.e. $b' \in D$ whenever $b \leq b'$ for some $b \in D$;
- $B - D$ is *downward closed under* $\leq$, i.e. $b' \in B - D$ whenever $b' \leq b$ for some $b \in B - D$;
- $D$ $(B - D)$ is *closed under lub and glb*, i.e. $D$ $(B - D)$ contains the lub and the glb of any non-empty subset of $D$ $(B - D)$.

The set of designated values $D$ will also be the set of truth values used to define paths.

Now, for any mv-Kripke structure $M = (Q, R, L)$ over a truth algebra $\mathcal{B}$ and a set of atomic propositions $AP$, and any set of designated values $D \subseteq B$ like above, we define a new 2-valued Kripke structure $M^D = (Q, R^D, L^D)$ such that:

- $R^D(q, q') = 1$ iff $R(q, q') \in D$, for any $q, q' \in Q$;
- $L^D(q)(p) = 1$ iff $L(q)(p) \in D$, for any $q \in Q$.

The main result of this reduction technique says that a formula $\phi$ has a designated truth value in $M$ if and only if it is 1 in $M^D$.

**Theorem 1.3** [73] Let $M = (Q, R, L)$ be an mv-Kripke structure like above and $D \subseteq B$ a set of designated values. Then, for any state (path) $CTL^*_+$ formula $\phi$ ($\psi$), any $q \in Q$ and any path $\pi \in \mathsf{Paths}(M, D)$ we have:

- $[\phi]^M_q \in D$ iff $[\phi]^{M^D}_q = 1$;
- $[\psi]^M_\pi \in D$ iff $[\psi]^{M^D}_\pi = 1$.

**Complexity** Clearly, the above theorem implies that the multi-valued model checking problem which questions the membership of the truth value of some formula to the set of designated values, has the same complexity as the 2-valued model checking problem.

### 1.3.2 Specialized techniques

#### Symbolic multi-valued $CTL$ model checking

In [16, 19, 17] a specialized model checking algorithm for multi-valued $CTL$ is proposed.

Multi-valued sets are sets for which the range of the membership function can be any quasi-boolean algebra and not the set $\{0, 1\}$ as in the classical set theory. Formally, if $\mathcal{B} = (B, \wedge, \vee, \neg)$ is a quasi-boolean algebra and $S$ a classical set, then a $\mathcal{B}$-valued set on $S$ is a total function from $S$ into $B$. The operations on multi-valued sets are defined as follows ($\mathcal{S}$ and $\mathcal{S}'$ are two $\mathcal{B}$-valued sets):

$$\begin{array}{rcl}
(\mathcal{S} \cap_{\mathcal{B}} \mathcal{S}')(x) & = & (\mathcal{S}(x) \wedge \mathcal{S}'(x)), \text{ for any } x \in S \cup S'; \\
(\mathcal{S} \cup_{\mathcal{B}} \mathcal{S}')(x) & = & (\mathcal{S}(x) \vee \mathcal{S}'(x)), \text{ for any } x \in S \cup S'; \\
\overline{\mathcal{S}}(x) & = & \neg\,(\mathcal{S}(x)), \text{ for any } x \in S; \\
\mathcal{S} \subseteq_{\mathcal{B}} \mathcal{S}' & = & \forall x\,(\mathcal{S}(x) \leq \mathcal{S}'(x)); \\
\mathcal{S} = \mathcal{S}' & = & \forall x\,(\mathcal{S}(x) = \mathcal{S}'(x)).
\end{array}$$

Straightforwardly, we can define $\mathcal{B}$-valued relations over two sets $S$ and $T$ as $\mathcal{B}$-valued sets on $S \times T$. If $\mathcal{R}$ is a $\mathcal{B}$-valued relation over $S$ and $T$ and $\mathcal{S}$

a $\mathcal{B}$-valued set on $S$ then we define the forward image of $\mathcal{S}$ under $\mathcal{R}$, $\overrightarrow{\mathcal{R}}(\mathcal{S})$, as follows:

$$\overrightarrow{\mathcal{R}}(\mathcal{S})(t) = \bigvee_{s \in S} (\mathcal{S}(s) \wedge \mathcal{R}(s,t)), \text{ for any } t \in T.$$

Analogously, if $\mathcal{R}$ is a $\mathcal{B}$-valued relation over $S$ and $T$ and $\mathcal{T}$ a $\mathcal{B}$-valued set on $T$ then we define the backward image of $\mathcal{T}$ under $\mathcal{R}$, $\overleftarrow{\mathcal{R}}(\mathcal{T})$, as follows:

$$\overleftarrow{\mathcal{R}}(\mathcal{T})(s) = \bigvee_{t \in T} (\mathcal{T}(t) \wedge \mathcal{R}(s,t)), \text{ for any } s \in S.$$

Algorithms for multi-valued model checking are now obtained by restating the $CTL$ operators in the $\mu$-calculus: usual operations on sets and relations are replaced by the multi-valued versions provided above. Moreover, as in classical symbolic model-checking, the authors of [16, 19, 17] use structures like *Multi-Valued Decision Diagrams* [109] and *Multi-Valued Binary-Terminal Decision Diagrams* [108] to efficiently manipulate multi-valued sets.

**Complexity** If $M$ is the complexity of 2-valued symbolic model checking, then the complexity of the multi-valued version is bounded by $t_{\mathcal{B}}\, M$, where $t_{\mathcal{B}}$ is the time needed to compute conjunctions and disjunctions of elements in $\mathcal{B}$. It is well known that this time is linear in the number of join-irreducible elements of $(B, \leq)$ (see, for example [37]).

### An automata approach to multi-valued $\mu$-calculus model checking

In [11] a multi-valued model checking algorithm for $\mu$-calculus using automata is proposed. As $CTL^*$ is expressible in $\mu$-calculus, this algorithm can be particularized for multi-valued $CTL^*$ model checking. It extends the approach for 2-valued model checking from [75] that uses alternating automata in the following way (we suppose $M$ is a 2-valued Kripke structure and $\phi$ a formula to be verified):

- constructs an alternating automaton $\mathcal{A}_\phi$ that accepts exactly the trees that satisfy $\phi$;
- constructs the product automaton $\mathcal{A}_{M,\phi} = M \times \mathcal{A}_{\mathcal{D},\phi}$;
- outputs "1" if the language of $\mathcal{A}_{M,\phi}$ is non-empty and "0", otherwise.

Although the approach in [11] applies to mv-Kripke structures with 2-valued transition predicates, this is not a restriction since any multi-valued model checking problem can be transformed into an equivalent one that uses

mv-Kripke structures with 2-valued transition predicates. For $\mu$-calculus, this is suggested but not formally proved in [11]. In the following we will provide a formal proof for the case of $CTL^*$ formulas. The main idea is that the truth value of a transition $R(q, q')$ in the initial model, is recorded in the state $q'$, in the new model. We emphasize that this will involve a blow-up of size $|B|$, where $\mathcal{B} = (B, \wedge, \vee, \neg)$ is the truth algebra under consideration.

Let $M = (Q, R, L)$ be an mv-Kripke structure over a truth algebra $\mathcal{B} = (B, \wedge, \vee, \neg)$ and some set of atomic propositions $AP$. We define the following structure $M' = (Q', R', L')$ over $\mathcal{B}$ and $AP \cup \{r\}$ such that $R'$ is 2-valued:

- $Q' = Q \times B$;
- $R'((q, b_1), (q', b_2)) = 1$ iff $R(q, q') = b_2$, for any $q, q' \in Q$ and $b_1, b_2 \in B$;
- $L'((q, b))(p) = L(q)(p)$, for any $q \in Q$ and $p \in AP$;
- $r$ is a new atomic proposition with $L'((q, b))(r) = b$, for any $q \in Q$ and $b \in B$.

The transformation above is useful when the set $D$ of truth values used to build paths is equal to $B - \{0\}$. In the general case, $Q' = Q \times D$ and $B$ is replaced with $D$. Moreover, notice that the set of truth values used to build path in $M'$ is $\{1\}$.

We define a transformation $T(\phi)$, for any $\phi$ a path or state $CTL^*$ formula, as follows (rules are given only for the minimal set of operators):

- $T(p) = p$, $T(\neg\varphi_1) = \neg T(\varphi_1)$ and $T(\varphi_1 \wedge \varphi_2) = T(\varphi_1) \wedge T(\varphi_2)$, for any $p \in AP$ and $\varphi_1$, $\varphi_2$ state formulas;
- $T(\forall\psi_1) = \forall\, T(\psi_1)$, for any $\psi_1$ a path formula;
- $T(\neg\psi_1) = \neg T(\psi_1)$, $T(\psi_1 \wedge \psi_2) = T(\psi_1) \wedge T(\psi_2)$, $T(\mathbf{X}\phi) = \mathbf{X}(T(\phi) \wedge r)$, $T(\overline{\mathbf{X}}\phi) = \overline{\mathbf{X}}(T(\phi) \wedge r)$ and $T(\phi\, \mathbf{U}\, \psi) = (T(\phi) \wedge \mathbf{X}r)\, \mathbf{U}\, T(\psi)$, for any $\psi_1$ and $\psi_2$ path formulas;

For any path $\pi$ of $M$ starting in $q$, we denote by $\pi'_b$ the path of $M'$ starting in $(q, b)$, for any $b \in B$ Notice that $\pi'_b$ is unique and $\pi'_b(i) = (\pi(i), b')$, where $R(\pi(i-1), \pi(i)) = b'$, for any $0 < i < |\pi|$.

**Theorem 1.4** Let $M$ and $M'$ be as above. Then,

$$[\phi]_q^M = [T(\phi)]_{(q,b)}^{M'} \text{ and } [\psi]_\pi^M = [T(\psi)]_{\pi'_b}^{M'}$$

for any $q \in Q$, state $CTL^*$ formula $\phi$, path $CTL^*$ formula $\psi$, and $b \in B$.

**Proof** We will prove the statements in the theorem by structural induction on the formulas $\phi$ and $\psi$. The following cases are to be considered:

- $\phi = p \in AP$. We have that $[\phi]^M_q = L(q)(p) = L'((q,b))(p)$, for any $b \in B$. Since $T(p) = p$ we obtain $[\phi]^M_q = [T(\phi)]^{M'}_{(q,b)}$;

- $\phi = \neg\phi_1$, where $\phi_1$ is a state formula. Then $[\phi]^M_q = \neg[\phi_1]^M_q$ and $T(\phi) = \neg T(\phi_1)$. Applying the induction hypothesis we get $[\phi_1]^M_q = [T(\phi_1)]^{M'}_{(q,b)}$, for any $b \in B$ and consequently, $[\phi]^M_q = [\neg T(\phi_1)]^{M'}_{(q,b)} = [T(\phi)]^{M'}_{(q,b)}$;

- $\phi = \phi_1 \wedge \phi_2$, where $\phi_1$ and $\phi_2$ are state formulas. Then $[\phi]^M_q = [\phi_1]^M_q \wedge [\phi_1]^M_q$ and $T(\phi) = T(\phi_1) \wedge T(\phi_2)$. Applying the induction hypothesis we get $[\phi_1]^M_q = [T(\phi_1)]^{M'}_{(q,b)}$ and $[\phi_2]^M_q = [T(\phi_2)]^{M'}_{(q,b)}$, for any $b \in B$. Consequently, $[\phi]^M_q = [T(\phi_1)]^{M'}_{(q,b)} \wedge [T(\phi_2)]^{M'}_{(q,b)} = [T(\phi)]^{M'}_{(q,b)}$;

- $\phi = \forall\psi$, where $\psi$ is a path formula. We have

  $$[\phi]^M_q = \bigwedge_{\pi \in \mathsf{Paths}(M,D,q)} [\psi]^M_\pi.$$

  Using $T(\phi) = \forall\, T(\psi)$ we obtain

  $$[T(\phi)]^{M'}_{(q,b)} = \bigwedge_{\rho \in \mathsf{Paths}(M',\{1\},(q,b))} [T(\psi)]^{M'}_\rho.$$

  By the definition of $M'$, any path $\rho \in \mathsf{Paths}(M', \{1\}, (q,b))$ is a path $\pi'_b$, for some $\pi \in \mathsf{Paths}(M, D, q)$. Now, applying the induction hypothesis, we get $[\psi]^M_\pi = [T(\psi)]^{M'}_{\pi'_b}$ and consequently, $[\phi]^M_q = [T(\phi)]^{M'}_{(q,b)}$;

- the case $\phi = \exists\phi_1$, where $\phi_1$ is a path formula, is similar to the previous one;

- the cases $\psi = \neg\psi_1$ and $\psi = \psi_1 \wedge \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas, are obtained as the similar cases involving state formulas;

- $\psi = \mathbf{X}\psi_1$, where $\psi_1$ is a path formula. We have $T(\psi) = \mathbf{X}(T(\psi_1) \wedge r)$ and

  $$[\psi]^M_\pi = (|\pi| > 1) \wedge R(\pi(0), \pi(1)) \wedge [\psi_1]^M_{\pi^1}.$$

  Clearly, $|\pi| > 1$ iff $|\pi_b| > 1$. Then,

  $$\begin{aligned} [T(\psi)]^{M'}_{\pi_b} &= [\mathbf{X}(T(\psi_1) \wedge r)]^{M'}_{\pi_b} \\ &= (|\pi_b| > 1) \wedge R'(\pi_b(0), \pi_b(1)) \wedge [T(\psi_1)]^{M'}_{\pi^1_b} \wedge [r]^{M'}_{\pi^1_b}. \end{aligned}$$

  By the definition of $\pi_b$ and $r$, we have $[r]^{M'}_{\pi^1_b} = R(\pi(0), \pi(1))$ which proves $[\psi]^M_\pi = [T(\psi)]^{M'}_{\pi_b}$;

- the case $\psi = \overline{\mathbf{X}}\psi_1$, where $\psi_1$ is a path formula is similar to the previous one;

- the case $\psi = \psi_1 \,\mathbf{U}\, \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas. We have $T(\psi) = (T(\psi_1) \wedge \mathbf{X}r) \,\mathbf{U}\, T(\psi_2)$ and

$$[\psi]^M_\pi = [\psi_2]^M_{\pi^0} \vee \bigvee_{0<i<|\pi|} ([\psi_2]^M_{\pi^i} \wedge \bigwedge_{0\leq j<i} ([\psi_1]^M_{\pi^j} \wedge R(\pi(j), \pi(j+1)))).$$

  Then,

$$\begin{aligned} [T(\psi)]^{M'}_{\pi_b} &= [(T(\psi_1) \wedge \mathbf{X}r) \,\mathbf{U}\, T(\psi_2)]^{M'}_{\pi_b} \\ &= [T(\psi_2)]^{M'}_{\pi^0_b} \vee \textstyle\bigvee_{0<i<|\pi_b|} \big([T(\psi_2)]^{M'}_{\pi^i_b} \wedge \\ &\quad \textstyle\bigwedge_{0\leq j<i}([T(\psi_1) \wedge \mathbf{X}r]^{M'}_{\pi^j_b} \wedge R'(\pi_b(j), \pi_b(j+1)))\big). \end{aligned}$$

  In the above, we use $R'(\pi_b(j), \pi_b(j+1)) = 1$,

$$[T(\psi_1) \wedge \mathbf{X}r]^{M'}_{\pi^j_b} = [T(\psi_1)]^{M'}_{\pi^j_b} \wedge [r]^{M'}_{\pi^{j+1}_b},$$

  and $[r]^{M'}_{\pi^{j+1}_b} = R(\pi(j), \pi(j+1)))$ to obtain $[\psi]^M_\pi = [T(\psi)]^{M'}_{\pi_b}$. □

The approach in [75] is extended to the multi-valued case by first extending the alternating automata. While in classic alternating automata, we describe successors through boolean expressions built up from states, boolean values, conjunction and disjunction, in *extended alternating automata* (EAA, for short) we describe successors through boolean expressions built up from states, truth values in $B$, conjunction and disjunction over $\mathcal{B}$. Moreover, in EAAs each accepting run has a truth value and now, we are not interested in the non-emptiness of the language but in the truth value of each accepting run.

The fundamental property of an EAA is that the set of truth values for which there exists an accepting run has a maximum value. This permits an algorithm which proceeds with minor differences as in the 2-valued case but, in the last step the answer for the truth value of $\phi$ in $M$ is the maximum value of an accepting run.

**Complexity** As computing the maximum value of an accepting run for the product automaton is proved to have the same complexity as checking it's non-emptiness, the complexity for $\mu$-calculus model checking remains the same even for the multi-valued case. However, in this case, the size of the problem will also include the size of $\mathcal{B}$.

# Chapter 2

# Multi-valued Abstractions

This chapter deals with abstractions that preserve properties expressed in logics under multi-valued interpretations given by truth algebras. The abstract system is obtained by applying equivalence relations and the predicate symbols of the logic are re-defined to work properly on equivalence classes.

We define abstractions of logical structures that preserve first order logic formulas, abstractions of multi-valued Kripke structures that preserve temporal logic formulas and abstractions of multi-agent multi-valued Kripke structures that preserve temporal logic of knowledge formulas. Moreover, we show how multi-valued abstractions can be computed from 2-valued abstractions provided that some requirements are satisfied.

Before giving abstractions of multi-valued Kripke structures, we provide a case study of using abstraction in the context of protection systems which model access control policies [63].

## 2.1 Interpretation policies

The abstraction techniques for multi-valued logics we introduce in this thesis extend the ones offered for Kleene's 3-valued logic in [111, 46, 45]. The abstract system is obtained by applying equivalence relations. Then, the predicate symbols of the logic are re-defined to work properly on equivalence classes. As an equivalence class may contain more than one element and each element leads to a truth value for each predicate, to redefine a predicate on an equivalence class comes down to define a policy of recombination of truth values from some given set.

**Definition 2.1** Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a truth algebra. An *interpretation policy* over $\mathcal{B}$ is any function $\alpha$ from $B$ into the set $\{\exists^S, \exists_a^{S'} | S, S' \in \mathcal{P}(B) -$

$\{\emptyset\}\}$ [1].

An interpretation policy $\alpha$ over $\mathcal{B}$ works as follows. Let $A$ be a non-empty set of elements, $p$ a unary predicate symbol, and $\mathcal{I}_p^A : A \to B$ an interpretation function which gives truth values to $p$ on each $a \in A$. Given an arbitrary and non-empty set $X \subseteq \mathcal{P}(A) - \{\emptyset\}$, we want to use $\alpha$ to define a new interpretation function $\mathcal{I}_p^X : X \to B$ as follows. For any $T \in X$, $\mathcal{I}_p^X(T)$ is the truth value $b \in B$ if one of the following properties is satisfied:

- if $\alpha(b) = \exists^S$, then

$$(\forall t \in T)(\mathcal{I}_p^A(t) \in S) \ \wedge \ (\exists t \in T)(\mathcal{I}_p^A(t) = b)$$

- if $\alpha(b) = \exists_a^S$, then

$$(\forall t \in T)(\mathcal{I}_p^A(t) \in S) \ \wedge \ (\exists t_1, t_2 \in T)(\mathcal{I}_p^A(t_1) \leq b \leq \mathcal{I}_p^A(t_2)).$$

It is quite clear that such a function $\mathcal{I}_p^X$ might not exists. It might happen that no policy $\alpha(b)$ can be applied to $T \in X$ or two distinct policies $\alpha(b)$ and $\alpha(b')$ can be applied to $T \in X$. When $\alpha$ allows to define an unique function $\mathcal{I}_p^X$ as above, then $\mathcal{I}_p^X$ will be called the *reinterpretation of* $\mathcal{I}_p^A$ *on* $X$ *according to* $\alpha$ and any element $\mathcal{I}_p^X(T)$ will be called the *truth value of* $p$ *on* $T$ *according to* $\alpha$ (it is also denoted by $p_\alpha^T$).

Therefore, an interpretation policy aims to redefine already existing interpretation functions. One may remark that a policy of the form $\alpha(b) = \exists^{\{b\}}$ says that a predicate $p$ is reinterpreted to $b$ on a non-empty subset $T$ if all the elements in $T$ evaluate the predicate $p$ to $b$. Sometimes, we will denote this by $\alpha(b) = \forall$.

**Example 2.1** In [111], an abstraction based verification technique has been proposed for abstract data types. The technique employs a first order logic under Kleene's 3-valued interpretation. The truth algebra is based on the lattice $(B = \{0, \perp, 1\}, \leq)$ with $0 \leq \perp \leq 1$. Three abstraction types were defined: $\forall\forall$-abstractions, $\forall\exists$-abstractions, and $\exists^{0,1}\forall$-abstractions.

It is easily seen that the reinterpretations of the predicate symbols for these abstractions are driven by the following interpretation policies:

| abstraction type [111] | interpretation policy | | |
|---|---|---|---|
| $\forall\forall$ | $\alpha(0) = \exists^{\{0\}}$ | $\alpha(\perp) = \exists_a^{\{0,\perp,1\}}$ | $\alpha(1) = \exists^{\{1\}}$ |
| $\forall\exists$ | $\alpha(0) = \exists^{\{0,\perp,1\}}$ | $\alpha(\perp) = \exists^{\{\perp,1\}}$ | $\alpha(1) = \exists^{\{1\}}$ |
| $\exists^{0,1}\forall$ | $\alpha(0) = \exists^{\{0\}}$ | $\alpha(\perp) = \exists^{\{0,\perp,1\}}$ | $\alpha(1) = \exists^{\{0,1\}}$ |

[1] The symbol $\exists$ in the definition of interpretation policies should not be confused with the symbol $\exists$ used in the syntax of the logics. In interpretation policies this symbol will always be accompanied by a superscript (and sometimes, a subscript).

As we have already said, an interpretation policy might not allow redefining an already existing interpretation function. Interestingly is that some interpretation policies allow redefining interpretation functions no matter what the domain of these functions is and no matter how these functions are defined. This is the case, for instance, of the policies in Example 2.1.

When an interpretation policy leads to exactly one reinterpretation no matter how the interpretation function is chosen, it will be called a *safe interpretation policy*. They can be characterized as follows.

**Theorem 2.1** An interpretation policy $\alpha : B \to \{\exists^S, \exists_a^{S'} | S, S' \in \mathcal{P}(B) - \{\emptyset\}\}$ is safe if and only if for any non-empty subset $S_T \subseteq B$ there exists an unique $b \in B$ such that one of the following two statements holds true:

- if $\alpha(b) = \exists^S$, then $b \in S_T \subseteq S$;
- if $\alpha(b) = \exists_a^S$, then $S_T \subseteq S$ and there are $b_1, b_2 \in S$ such that $b_1 \leq b \leq b_2$.

**Proof** It is clear that if $\alpha$ is safe then there is an unique $b \in B$ such that one of the two statements holds true.

Conversely, assume that $\alpha$ satisfies the property in the theorem. Let $p$ be a predicate symbol, $A$ be a non-empty set, and $\mathcal{I}_p^A : A \to B$ be an interpretation of $p$ over $A$. Let $X \subseteq \mathcal{P}(A) - \{\emptyset\}$. We will show that we can define an unique reinterpretation $\mathcal{I}_p^X$ of $\mathcal{I}_p^A$ on $X$ according to $\alpha$.

Let $T \in X$, and let $S_T = \{\mathcal{I}_p^A(t) | t \in T\}$. Clearly, $S_T$ is non-empty. Then, there exists an unique $b \in B$ such that one of the two properties in the theorem holds true. If we define $\mathcal{I}_p^X(T) = b$ then one can easily see that this is a consistent definition of a function and it is the unique one which satisfies the interpretation policy $\alpha$. □

## 2.2 First-Order Logic

### 2.2.1 Abstractions and Preservation Results

Logical structures, used to model systems, assign meanings to logical signatures by associating a set of data to each kind and a multi-valued predicate to each predicate symbol. Abstractions of logical structures are captured by using equivalences and interpretation policies. The equivalence classes represent sets of elements that are treated as a whole and the interpretation policy is used to obtain the truth value of the predicates on equivalence classes.

If $K$ is a non-empty set of kinds, a *$K$-kinded binary relation* on a $K$-kinded set $S$ is a $K$-indexed family $\rho = (\rho_k | k \in K)$ such that $\rho_k$ is a binary

relation on $S_k$. $\rho$ is an *equivalence relation* on $S$ if each $\rho_k$ is an equivalence relation on $S_k$.

**Definition 2.2** Let $\mathcal{S} = (S, \Sigma_L^S)$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ a $K$-kinded equivalence on $S$ and $\alpha$ an interpretation policy over $\mathcal{B}$. An *$\alpha$-abstraction of $\mathcal{S}$ by $\rho$* is a $(\mathcal{B}, \Sigma_L)$-logical structure $\mathcal{S}' = (S/\rho, \Sigma_L^{S'})$ such that:

- $S/\rho = (S_k/\rho_k | k \in K)$;
- $p^{\mathcal{S}'}([a_1]_{\rho_{k_1}}, \ldots, [a_m]_{\rho_{k_m}})$ is the value of $p$ over the set

$$\{(u_1, \ldots, u_m) | u_i \in [a_i]_{\rho_{k_i}}, 1 \leq i \leq m\}$$

  according to $\alpha$, for any predicate $p$ of type $k_1 \ldots k_m$ with $m \geq 1$, and $a_i \in S_{k_i}$, $1 \leq i \leq m$.

In order to be useful, an abstraction should be *property preserving* with respect to a specific set of properties. In the following we provide three types of preservation results:

- *$\geq$-preservation with respect to a set of properties $P$ and two truth values* $b, b' \in B$ means that a given property $\phi \in P$ is evaluated to a truth value greater than or equal to $b'$ in the concrete system whenever it is evaluated to a truth value grater than or equal to $b$ in the abstract system;
- *$\leq$-preservation with respect to a set of properties $P$ and two truth values* $b, b' \in B$ means that a given property $\phi \in P$ is evaluated to a truth value less than or equal to $b'$ in the concrete system whenever it is evaluated to a truth value less than or equal to $b$ in the abstract system;
- *$=$-preservation results with respect to a set of properties $P$ and a set of truth values $B'$* means that a given property $\phi \in P$ is evaluated to a truth value $b \in B'$ in the concrete system whenever it is evaluated to $b$ in the abstract system;

**Remark 2.1** If $B$ is the classic two-valued truth algebra, the $=$-preservation include three forms of property preservation frequently found in the literature [34, 113, 111]:

- *strong-preservation*: an abstraction is strongly preserving if a set of properties with truth values true or false in the abstract system has corresponding properties in the concrete system with the same truth values;

- *weak-preservation*: an abstraction is weakly preserving if a set of properties true in the abstract system has corresponding properties in the concrete system that are also true;

- *error-preservation*: an abstraction is error preserving if a set of properties false in the abstract system has corresponding properties in the concrete system that are also false.

One of the advantages of Definition 2.2 is that with the same equivalence we can prove more than one property just by changing the interpretation policy.

We shall derive in the following, several preservation results. We begin by two technical results. Given an equivalence $\rho$ on $S$ and two assignments $\gamma \in \Gamma(X, \mathcal{S}')$ and $\gamma' \in \Gamma(X, \mathcal{S})$, we write $\gamma' \in \gamma$ whenever $\gamma'(x) \in \gamma(x)$, for all $x \in X$.

**Lemma 2.1** Let $\mathcal{S}$ be a $(\mathcal{B}, \Sigma_L)$-logical structure like above, $\rho$ an equivalence on $S$ and $\mathcal{S}'$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, for some interpretation policy $\alpha$. Then:

(1) $\gamma' \in \Gamma(X, \mathcal{S})$, for any $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\gamma' \in \gamma$;

(2) for any $\gamma' \in \Gamma(X, \mathcal{S})$ there exists $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\gamma' \in \gamma$.

**Proof** (1) follows directly from definitions.

(2) Given an assignment $\gamma'$ into $\mathcal{S}$, define the assignment $\gamma$ into $\mathcal{S}'$ by $\gamma(x) = [\gamma'(x)]$, for all $x$. Clearly, $\gamma' \in \gamma$. □

**Lemma 2.2** Let $\mathcal{S}$, $\rho$, and $\mathcal{S}'$ like above, and $\gamma$ be an assignment into $\mathcal{S}'$. Then,

$$\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = p^{\mathcal{S}'}([\gamma_1(x_1)], \ldots, [\gamma_m(x_m)]),$$

for any $\gamma_1, \ldots, \gamma_m \in \gamma$ and atomic formula $p(x_1, \ldots, x_m)$.

**Proof** Directly from the definition of $\mathcal{I}_{\mathcal{S}'}$. □

We have to remark that Lemmatta 2.1 and 2.2 do not depend on the interpretation policy. As a result they hold true for any abstraction.

**Theorem 2.2** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^T, \exists_a^T, \forall \mid T \subseteq\uparrow b'\}$, for all $x \geq b$, then:

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \geq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \geq b'),$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$. Moreover, if

$$\vee B' \geq b \;\Rightarrow\; (\exists x \in B')(x \geq b), \text{ for any } B' \subseteq B,$$

then

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \geq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \geq b'),$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$.

**Proof** To prove the first part, we proceed by structural induction on $\phi$. We consider the following cases:

- $\phi = p(x_1, \ldots, x_m)$ is an atomic formula. Let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) \geq b$. Now, let $\gamma' \in \Gamma(X, \mathcal{S})$ such that $\gamma' \in \gamma$. We apply Lemma 2.2 for $\gamma_1 = \cdots = \gamma_m = \gamma'$ and obtain

  $$\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = p^{\mathcal{S}'}([\gamma'(x_1)], \ldots, [\gamma'(x_m)]) \geq b,$$

  which by the conditions on $\alpha$ implies $p^{\mathcal{S}}(\gamma'(x_1), \ldots, \gamma'(x_m)) \geq b'$. Consequently, $\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \geq b'$;

- $\phi = \phi_1 \wedge \phi_2$. Assume that $\phi_1$ and $\phi_2$ satisfy the property and let $\gamma \in \Gamma(X, \mathcal{S}')$ be an assignment such that $\mathcal{I}_{\mathcal{S}'}(\phi_1 \wedge \phi_2)(\gamma) \geq b$. This implies $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) \geq b$ and $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \geq b$. By the induction hypothesis, we obtain $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \geq b'$ and $\mathcal{I}_{\mathcal{S}}(\phi_2)(\gamma') \geq b'$, for any $\gamma' \in \gamma$. Therefore, $\mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2)(\gamma') \geq b'$, for any $\gamma' \in \gamma$;

- $\phi = (\forall x)\phi_1$ with $x \in X_k$. Assume that $\phi_1$ satisfies the property and let $\gamma \in \Gamma(X, \mathcal{S}')$ be an assignment such that $\mathcal{I}_{\mathcal{S}'}((\forall x)\phi_1)(\gamma) \geq b$. This implies $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma[x/a]) \geq b$, for any $a \in S_k/\rho_k$. By the induction hypothesis, we obtain that $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma'') \geq b'$, for any $\gamma'' \in \gamma[x/a]$. Consequently, $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma'[x/a']) \geq b'$, for any $\gamma' \in \gamma$ and $a' \in a$, and, $\mathcal{I}_{\mathcal{S}}((\forall x)\phi_1)(\gamma') \geq b'$, for any $\gamma' \in \gamma$.

For the second part, two more cases are to be added to the proof above:

- $\phi = \phi_1 \vee \phi_2$. Assume that $\phi_1$ and $\phi_2$ satisfy the property and let $\gamma \in \Gamma(X, \mathcal{S}')$ be an assignment such that $\mathcal{I}_{\mathcal{S}'}(\phi_1 \vee \phi_2)(\gamma) \geq b$. By the condition on the truth algebra $\mathcal{B}$, we obtain that $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) \geq b$ or $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \geq b$, and by the induction hypothesis we obtain $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \geq b'$, for any $\gamma' \in \gamma$ or $\mathcal{I}_{\mathcal{S}}(\phi_2)(\gamma') \geq b'$, for any $\gamma' \in \gamma$. Thus, $\mathcal{I}_{\mathcal{S}}(\phi_1 \vee \phi_2)(\gamma') \geq b'$, for any $\gamma' \in \gamma$;

- $\phi = (\exists x)\phi_1$, with $x \in X_k$. Assume that $\phi_1$ satisfies the property and let $\gamma \in \Gamma(X, \mathcal{S}')$ be an assignment such that $\mathcal{I}_{\mathcal{S}'}((\exists x)\phi_1)(\gamma) \geq b$. By the condition on the truth algebra $\mathcal{B}$, this implies that there exists $a \in S_k/\rho_k$ such that $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma[x/a]) \geq b$, and by the induction hypothesis, we obtain $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma'') \geq b'$, for any $\gamma'' \in \gamma[x/a]$. Thus, there exists $a' \in a$ such that $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma'[x/a']) \geq b'$, for any $\gamma' \in \gamma$, and, $\mathcal{I}_{\mathcal{S}}((\exists x)\phi_1)(\gamma') \geq b'$, for any $\gamma' \in \gamma$. □

**Corollary 2.1** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^T, \exists_a^T, \forall \mid T \subseteq\uparrow b'\}$, for all $x \geq b$, then:

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b',$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L, X)$. Moreover, if

$$\vee B' \geq b \;\Rightarrow\; (\exists x \in B')(x \geq b), \text{ for any } B' \subseteq B,$$

then

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b',$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$.

**Proof** Let $\phi$ be a formula in $\mathcal{L}^{\{\wedge\}}(\Sigma_L, X)$ such that $[\phi]^{\mathcal{S}'} \geq b$. Then, $\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \geq b$, for any $\gamma \in \Gamma(X, \mathcal{S}')$. By Lemma 2.1, for any $\gamma' \in \Gamma(X, \mathcal{S})$ there exists $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\gamma' \in \gamma$. Consequently, we can use Theorem 2.2 to obtain $[\phi]^{\mathcal{S}} \geq b'$. The second part of the theorem follows a similar line. □

We exemplify the use of the $\geq$-preservation result in Corollary 2.1.

**Example 2.2** Let $K$ be a set of kinds containing the kind *nat* for the set of natural numbers and $(\mathcal{B}, \Sigma_L)$ be a $K$-kinded logical signature where $\mathcal{B}$ is the truth algebra in Figure 1.1(e), $p \in \Sigma_{L,nat}$, and $q \in \Sigma_{L,nat\,nat}$. Moreover, consider $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ a $(\mathcal{B}, \Sigma_L)$-logical structure such that:

- $S_{nat} = \mathbf{N}$;
- $p^{\mathcal{S}}$ is defined by:

$$p^{\mathcal{S}}(x) = \begin{cases} 1M, & \text{if } x\%4 = 0; \\ 10, & \text{if } x\%4 = 1; \\ 11, & \text{if } x\%4 = 2; \\ 1M, & \text{if } x\%4 = 3; \end{cases}$$

- $q^{\mathcal{S}}$ is defined by:

$$q^{\mathcal{S}}(x,y) = \begin{cases} MM, & \text{if } (x+y)\%4 = 0; \\ M1, & \text{if } (x+y)\%4 = 1; \\ M0, & \text{if } (x+y)\%4 = 2; \\ MM, & \text{if } (x+y)\%4 = 3; \end{cases}$$

Suppose we are interested in the truth value of the formula $\phi_1 = p(x) \wedge p(y) \wedge q(x,y)$. We can find a lower bound for this truth value by applying an $\alpha$-abstraction of $\mathcal{S}$ by the equivalence $\rho$ given by

$$x \, \rho_{nat} \, y \text{ iff } x \text{ and } y \text{ have the same parity.}$$

The interpretation policy $\alpha$ is chosen such that

| Truth values | $1M$ | $10$ | $M0$ | $MM$ | $11$ | $M1$ |
|---|---|---|---|---|---|---|
| The policy $\alpha$ | $\exists^{\{1M,11\}}$ | $\exists^{\{1M,10\}}$ | $\exists^{\{M0,MM\}}$ | $\exists^{\{MM,M1\}}$ | $\forall$ | $\forall$ |

Let $\mathcal{S}'$ be the abstraction above. Since $|S_{nat}/\rho| = 2$ we can easily prove that $[\phi_1]^{\mathcal{S}'} = M0$. Moreover, we have $\alpha(x) \in \{\exists^T, \exists^T_a, \forall \mid T \subseteq\uparrow M0\}$, for all $x \geq M0$ and, consequently, $[\phi_1]^{\mathcal{S}} \geq M0$, by Corollary 2.1.

**Theorem 2.3** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^{S'}, \exists^{S'}_a, \forall \mid S' \subseteq\downarrow b'\}$, for all $x \leq b$, then:

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \leq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq b'),$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$. Moreover, if

$$\wedge B' \leq b \;\Rightarrow\; (\exists x \in B')(x \leq b), \text{ for any } B' \subseteq B,$$

then

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \leq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq b'),$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$.

**Proof** This is similar to the proof of Theorem 2.2. □

**Corollary 2.2** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^{S'}, \exists^{S'}_a, \forall \mid S' \subseteq\downarrow b'\}$, for all $x \leq b$, then:

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b',$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L, X)$. Moreover, if

$$\wedge B' \leq b \;\Rightarrow\; (\exists x \in B')(x \leq b), \text{ for any } B' \subseteq B,$$

then

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b',$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$.

**Proof** This is similar to the proof of Corollary 2.1 (but using Theorem 2.3). □

**Example 2.3** Let $K$ be a set of kinds containing the kind *nat* for the set of natural numbers. We consider a $K$-kinded logical signature $(\mathcal{B}, \Sigma_L)$ where $\mathcal{B}$ is the truth algebra in Figure 1.1(h), $p \in \Sigma_{L,nat}$, and $q \in \Sigma_{L,nat\,nat}$. Moreover, let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure such that:

- $S_{nat} = \mathbf{N}$;
- $p^{\mathcal{S}}(x) = SH$ if $x$ is even and $p^{\mathcal{S}}(x) = 1$ if $x$ is odd;
- $q^{\mathcal{S}}(x, y) = DC$ if $x + y$ is even and $q^{\mathcal{S}}(x, y) = SH$ if $x + y$ is odd.

Consider the formula

$$\phi_2 = (\forall x)(\exists y)(p(x) \vee q(x, y)).$$

An upper bound for the truth value of $\phi_2$ can be found by applying an $\alpha$-abstraction of $\mathcal{S}$ induced by the equivalence relation $\rho$ given by:

$$x\, \rho_{nat}\, y \text{ iff } x \text{ and } y \text{ have the same parity},$$

for any $x, y \in \mathbf{N}$. The interpretation policy $\alpha$ can be chosen in various ways. For example, suppose that $\alpha(DC) = \alpha(SH) = \alpha(1) = \forall$.

If $\mathcal{S}'$ is the $\alpha$-abstraction of $\mathcal{S}$ by $\rho$ then the value of $[\phi_2]^{\mathcal{S}'}$ can be obtained easily because $S_{nat}/\rho = \{[0]_\rho, [1]_\rho\}$ has only two elements. More exactly, $[\phi_2]^{\mathcal{S}'}$ equals

$$\begin{array}{l}\big((p([0]_\rho) \vee q([0]_\rho, [0]_\rho)) \vee (p([0]_\rho) \vee q([0]_\rho, [1]_\rho))\big) \wedge \\ \big((p([1]_\rho) \vee q([1]_\rho, [0]_\rho)) \vee (p([1]_\rho) \vee q([1]_\rho, [1]_\rho))\big) = SH.\end{array}$$

Now, since $\alpha(x) = \forall$, for all $x \leq SH$, and

$$\wedge B' \leq SH \;\Rightarrow\; (\exists x \in B')(x \leq SH),$$

we obtain $[\phi_2]^{\mathcal{S}} \leq SH$, by Corollary 2.2.

**Theorem 2.4** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq \uparrow b \cap \downarrow x\}$, for all $x > b$, then:

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \geq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(b \leq \mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi)(\gamma)) \text{ and}$$
$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) = b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') = b),$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$.

**Proof** We proceed by simultaneous structural induction on $\phi$. The following cases are to be discussed:

- $\phi = p(x_1, \ldots, x_m)$ is an atomic formula. Let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) \geq b$. Now, let $\gamma' \in \Gamma(X, \mathcal{S})$ such that $\gamma' \in \gamma$. We apply Lemma 2.2 for $\gamma_1 = \cdots = \gamma_m = \gamma'$ and obtain

  $$\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = p^{\mathcal{S}'}([\gamma'(x_1)], \ldots, [\gamma'(x_m)]) \geq b,$$

  which implies $b \leq p^{\mathcal{S}}(\gamma'(x_1), \ldots, \gamma'(x_m)) \leq p^{\mathcal{S}'}(\gamma(x_1), \ldots, \gamma(x_m))$. Consequently, $b \leq \mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi)(\gamma)$.

  Now, let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = b$. We can use in the same way Lemma 2.2 and $\alpha(b) = \forall$ to obtain $\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') = b$, for any $\gamma' \in \gamma$;

- $\phi = \phi_1 \wedge \phi_2$. Assume that $\phi_1$ and $\phi_2$ satisfy the properties and let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(\phi_1 \wedge \phi_2)(\gamma) \geq b$. This implies $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) \geq b$ and $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \geq b$. By the induction hypothesis we obtain $b \leq \mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma)$ and $b \leq \mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma)$, for any $\gamma' \in \gamma$, which further implies $b \leq \mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi_1 \wedge \phi_2)(\gamma)$.

  Now, let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(\phi_1 \wedge \phi_2)(\gamma) = b$. We consider two cases:

  - $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) = b$ and $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \geq b$. By the induction hypothesis, the first statement implies $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') = b$, for any $\gamma' \in \gamma$ while the latter implies $\mathcal{I}_{\mathcal{S}}(\phi_2)(\gamma') \geq b$, for any $\gamma' \in \gamma$ and, consequently, $\mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2)(\gamma') = b$, for any $\gamma' \in \gamma$;
  - $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) \neq b$ and $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \neq b$. Again, by the induction hypothesis, we obtain $b \leq \mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma)$ and $b \leq \mathcal{I}_{\mathcal{S}}(\phi_2)(\gamma') \leq \mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma)$, for any $\gamma' \in \gamma$ which implies $\mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2)(\gamma') = b$, for any $\gamma' \in \gamma$. □

**Corollary 2.3** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq \uparrow b \cap \downarrow x\}$, for all $x > b$, then:

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; b \leq [\phi]^{\mathcal{S}} \leq [\phi]^{\mathcal{S}'} \text{ and}$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Sigma_L, X)$.

**Proof** This is similar to the proof of Corollary 2.1 (but using Theorem 2.4). □

**Theorem 2.5** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq \uparrow x \cap \downarrow b\}$, for all $x < b$, then:

$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \leq b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \leq \mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq b), \text{ and}$$
$$\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) = b \;\Rightarrow\; (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') = b),$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$.

**Proof** It follows the same line as the proof of Theorem 2.4. □

**Corollary 2.4** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq \uparrow x \cap \downarrow b\}$, for all $x < b$, then:

$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}'} \leq [\phi]^{\mathcal{S}} \leq b \text{ and}$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Sigma_L, X)$.

**Proof** This is similar to the proof of Corollary 2.1 (but using Theorem 2.5). □

**Theorem 2.6** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If:

1. $\alpha(b) = \forall$;
2. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\uparrow b\}$, for all $x > b$;
3. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\downarrow b\}$, for all $x < b$;
4. for any $B' \subseteq B$, $\wedge B' \leq b$ implies that there exists $x \in B'$ such that $x \leq b$;
5. for any $B' \subseteq B$, $\vee B' \geq b$ implies that there exists $x \in B'$ such that $x \geq b$;
6. for any $B' \subseteq B$, $\wedge B' = b$ implies $b \in B'$;
7. for any $B' \subseteq B$, $\vee B' = b$ implies $b \in B'$;

then

$$\begin{aligned}
\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \geq b &\Rightarrow (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \geq b),\\
\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) \leq b &\Rightarrow (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') \leq b),\\
\mathcal{I}_{\mathcal{S}'}(\phi)(\gamma) = b &\Rightarrow (\forall \gamma' \in \gamma)(\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') = b),
\end{aligned}$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$ and $\gamma \in \Gamma(X, \mathcal{S}')$.

**Proof** The first and the second statement in the theorem are obtained by taking $b' = b$ in Theorem 2.2 and 2.3, respectively. For the third statement, we proceed by structural induction on $\phi$ and consider the following cases:

- $\phi = p(x_1, \ldots, x_m)$ is an atomic formula. Let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = b$ and $\gamma' \in \Gamma(X, \mathcal{S})$ with $\gamma' \in \gamma$. We apply Lemma 2.2 for $\gamma_1 = \cdots = \gamma_m = \gamma'$ and obtain

  $$\mathcal{I}_{\mathcal{S}'}(p(x_1, \ldots, x_m))(\gamma) = p^{\mathcal{S}'}([\gamma'(x_1)], \ldots, [\gamma'(x_m)]) = b,$$

  which implies $p^{\mathcal{S}}(\gamma'(x_1), \ldots, \gamma'(x_m)) = b$, by $\alpha(b) = \forall$. Consequently, $\mathcal{I}_{\mathcal{S}}(\phi)(\gamma') = b$, for any $\gamma' \in \gamma$;

- $\phi = \phi_1 \wedge \phi_2$. Assume that $\phi_1$ and $\phi_2$ satisfy the properties and let $\gamma \in \Gamma(X, \mathcal{S}')$ such that $\mathcal{I}_{\mathcal{S}'}(\phi_1 \wedge \phi_2)(\gamma) = b$. By 6, we can suppose without any loss of generality that $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma) = b$ and $\mathcal{I}_{\mathcal{S}'}(\phi_2)(\gamma) \geq b$. By the induction hypothesis we obtain $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') = b$ and $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma') \geq b$, for any $\gamma' \in \gamma$, which further implies $\mathcal{I}_{\mathcal{S}}(\phi_1 \wedge \phi_2)(\gamma') = b$, for any $\gamma' \in \gamma$;

- $\phi = \phi_1 \vee \phi_2$. This is similar to the previous case;

- $\phi = (\forall x)\phi_1$ with $x \in X_k$. Assume that $\phi_1$ satisfies the property, and let $\gamma \in \Gamma(X, \mathcal{S}')$ be an assignment such that $\mathcal{I}_{\mathcal{S}'}((\forall x)\phi_1)(\gamma) = b$. This implies $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma[x/a]) \geq b$, for any $a \in S_k/\rho_k$ and, by 6, $\mathcal{I}_{\mathcal{S}'}(\phi_1)(\gamma[x/a']) = b$, for some $a' \in S_k/\rho_k$. By the induction hypothesis, we obtain that $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma_1) \geq b$, for any $\gamma_1 \in \gamma[x/a]$, and $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma_2) = b$, for any $\gamma_2 \in \gamma[x/a']$. Consequently, $\mathcal{I}_{\mathcal{S}}(\phi_1)(\gamma'[x/a_1]) \geq b$ and $\mathcal{I}_{\mathcal{S}}((\forall x)\phi_1)(\gamma'[x/a_2]) = b$, for any $\gamma' \in \gamma$, $a_1 \in S_k$ and $a_2 \in a'$. Therefore, $\mathcal{I}_{\mathcal{S}}((\forall x)\phi_1)(\gamma') = b$, for any $\gamma' \in \gamma$;

- $\phi = (\exists x)\phi_1$. This is similar to the previous case. □

**Corollary 2.5** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, and $b$ a truth value in $B$. If:

1. $\alpha(b) = \forall$;
2. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\uparrow b\}$, for all $x > b$;
3. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\downarrow b\}$, for all $x < b$;
4. for any $B' \subseteq B$, $\wedge B' \leq b$ implies that there exists $x \in B'$ such that $x \leq b$;
5. for any $B' \subseteq B$, $\vee B' \geq b$ implies that there exists $x \in B'$ such that $x \geq b$;
6. for any $B' \subseteq B$, $\wedge B' = b$ implies $b \in B'$;
7. for any $B' \subseteq B$, $\vee B' = b$ implies $b \in B'$;

then

$$[\phi]^{\mathcal{S}'} \geq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \geq b,$$
$$[\phi]^{\mathcal{S}'} \leq b \;\Rightarrow\; [\phi]^{\mathcal{S}} \leq b,$$
$$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b,$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$.

**Proof** This is similar to the proof of Corollary 2.1 (but using Theorem 2.6). □

All preservation results in [111] (for Kleene's 3-valued logic) are particular cases of Corollary 2.5.

**Corollary 2.6** Let $\leq$ be the partial order on $B = \{0, \perp, 1\}$ given by $0 \leq \perp \leq 1$ and let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be the corresponding truth algebra. Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure, $\rho$ an equivalence on $S$, $\alpha$ an interpretation policy over $\mathcal{B}$, and $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$. Then, the following properties hold:

- if $\alpha(0) = \forall$, $\alpha(\perp) = \exists^{\{0,\perp,1\}}$, and $\alpha(1) = \exists^{\{0,1\}}$ then

  $$[\phi]^{\mathcal{S}'} = 0 \;\Rightarrow\; [\phi]^{\mathcal{S}} = 0, \text{ for any } \phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X).$$

  That is, $\mathcal{S}'$ is error-preserving with respect to formulas in $\mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$;

- if $\alpha(0) = \exists^{\{0,\perp,1\}}$, $\alpha(\perp) = \exists^{\{\perp,1\}}$, and $\alpha(1) = \forall$, then

  $$[\phi]^{\mathcal{S}'} = 1 \;\Rightarrow\; [\phi]^{\mathcal{S}} = 1, \text{ for any } \phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X).$$

  That is, $\mathcal{S}'$ is weak-preserving with respect to formulas in $\mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X)$;

- if $\alpha(0) = \forall$, $\alpha(\perp) = \exists_a^{\{0,\perp,1\}}$, and $\alpha(1) = \forall$, then

  $$[\phi]^{\mathcal{S}'} = b \;\Rightarrow\; [\phi]^{\mathcal{S}} = b, \text{ for any } \phi \in \mathcal{L}(\Sigma_L, X) \text{ and } b \in \{0, 1\}.$$

  That is, $\mathcal{S}'$ is strong-preserving with respect to formulas in $\mathcal{L}(\Sigma_L, X)$;

### 2.2.2 Multi-valued Abstraction Through 2-valued Abstraction

In this section we show how multi-valued abstractions can be computed from 2-valued abstractions, provided that some requirements are satisfied. In order to simplify the presentation, we will consider only 2-valued abstractions based on the two possible safe interpretation policies $\alpha_1$ and $\alpha_2$ over $\mathcal{B}_2$ ($\mathcal{B}_2$ is the truth algebra with two elements):

- $\alpha_1(0) = \exists$ and $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \forall$ and $\alpha_2(1) = \exists$.

We may remark that $\alpha_1$ corresponds to the interpretation by *under-approximation* (i.e. $p([a]_\rho) = 1$ iff $p(a_1) = 1$, for all $a_1 \in [a]_\rho$) and $\alpha_2$ corresponds to the interpretation by *over-approximation* (i.e. $p([a]_\rho) = 1$ iff $p(a_1) = 1$, for some $a_1 \in [a]_\rho$).

**Definition 2.3** Let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}_2, \Sigma_L)$-logical structure and $\rho$ a $K$-kinded equivalence on $S$. A *$\mathcal{P}$-abstraction of $\mathcal{S}$ by $\rho$*, where $\mathcal{P} \subseteq \Sigma_L$, is a $(\mathcal{B}_2, \Sigma_L)$-logical structure $\mathcal{S}' = (S/\rho, \Sigma_L^{\mathcal{S}'})$ such that:

- $S/\rho = (S_k/\rho_k | k \in K)$;
- $p^{\mathcal{S}'}([a_1]_{\rho_{k_1}}, \ldots, [a_m]_{\rho_{k_m}})$ is the value of $p$ over the set

$$\{(u_1, \ldots, u_m) | u_i \in [a_i]_{\rho_{k_i}}, 1 \leq i \leq m\}$$

  according to $\alpha_1$ if $p \in \mathcal{P}$ and according to $\alpha_2$ if $p \notin \mathcal{P}$, for any predicate $p$ of type $k_1 \ldots k_m$ with $m \geq 1$, and $a_i \in S_{k_i}$, $1 \leq i \leq m$.

The passing from 2-valued abstractions to multi-valued abstractions follows two steps:

- first, we represent multi-valued predicates by sets of 2-valued predicates induced by some new partial order $\leq'$ on $B$. Using this representation, a $(\mathcal{B}_2, \Sigma'_L)$-logical structure $\mathcal{S}_{\leq'}$ is associated to any $(\mathcal{B}, \Sigma_L)$-logical structure $\mathcal{S}$;
- second, if $\mathcal{S}'_{\leq'}$ is a $\mathcal{P}$-abstraction of $\mathcal{S}_{\leq'}$ by some equivalence $\rho$, then we extract a interpretation policy $\alpha$ over $\mathcal{B}$ such that the $(\mathcal{B}, \Sigma_L)$-logical structure $\mathcal{S}'$ associated to $\mathcal{S}'_{\leq'}$ is an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$.

Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a truth algebra over the lattice $(B, \leq)$. We consider a new partial order $\leq'$ on $B$ with the following properties [2]:

- $(B, \leq')$ is an *inf-complete lattice*, i.e. any subset $B' \subseteq B$ has a greatest lower bound. We denote by $0'$ its smallest element;
- for any $x \in B$ there exists a unique $y \in B$ such that $x \succ' y$.

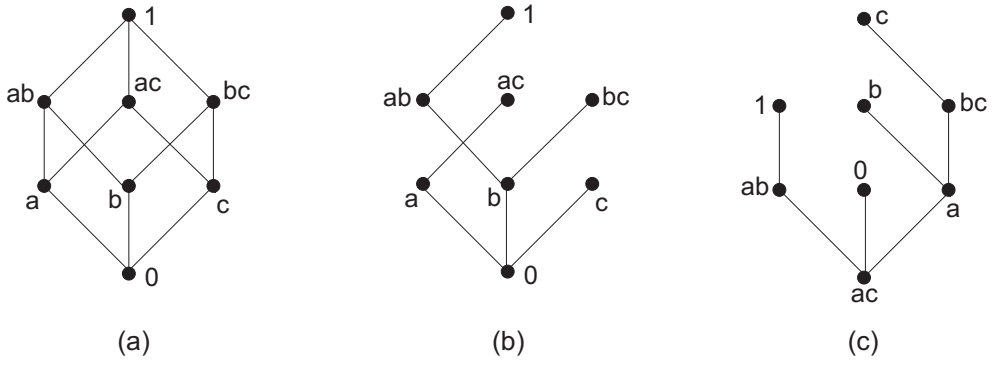


Figure 2.1: A lattice and two possible orders $\leq'$ for it.

[2] As usual, if $\leq'$ is a partial order on $B$, then $<'=\leq' - \{(a,a) | a \in B\}$, $>'$ is the inverse of $<'$, and $x \succ' y$ if $x >' y$ and there exists no $c \in B$ such that $x >' c >' y$.

**Example 2.4** A lattice representing the inclusion order between subsets of $\{a, b, c\}$ (0 stands for the empty set and 1 for the full set $\{a, b, c\}$) is given in Figure 2.1(a). Two possible orders $\leq'$ for it are represented in Figure 2.1(b) and Figure 2.1(c).

Now, let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $K$-kinded $(\mathcal{B}, \Sigma_L)$-logical structure and $\leq'$ a partial order on $B$ as above. We will associate to $\mathcal{S}$ a $K$-kinded $(\mathcal{B}_2, \Sigma'_L)$-logical structure $\mathcal{S}_{\leq'} = (S, \Sigma_L'^{\mathcal{S}_{\leq'}})$ in the following way:

- for each predicate symbol $p \in \Sigma_L$ we include in $\Sigma'_L$ a set of predicates $\{p_b \mid b \in B - \{0'\}\}$ of the same type with $p$;
- $p_b^{\mathcal{S}_{\leq'}}(\vec{a}) = 1$ if $p^{\mathcal{S}}(\vec{a}) \geq' b$, for any $\vec{a} \in S_w$, where $w$ is the type of $p$. In words, $p_b$ is true for elements on which $p$ gets truth values greater than or equal to $b$ (with respect to $\leq'$).

**Example 2.5** Let $K = \{k\}$ be a set of kinds and $(\mathcal{B}, \Sigma_L)$ a logical signature, where $\mathcal{B}$ is the truth algebra in Figure 2.1(a) and $\Sigma_{L,k} = \{p\}$. Moreover, let $\mathcal{S} = (S, \Sigma_L^{\mathcal{S}})$ be a $(\mathcal{B}, \Sigma_L)$-logical structure such that the support of the kind $k$ is the language $\{a^m b^n c^p \mid m, n, p \in \mathbf{N}\}$, and $p^{\mathcal{S}}(w)$ is the truth value representing the maximal set of symbols that appear in $w$ (the value of $p$ over the empty word $\epsilon$ is 0, and the value of $p$ over the words containing all symbols is 1). For example, $p^{\mathcal{S}}(b^7) = b$, $p^{\mathcal{S}}(a^2 b) = ab$ and $p^{\mathcal{S}}(ab^3c^8) = 1$.

Based on the partial order $\leq'$ Figure 2.1(c), we can associate to $\mathcal{S}$ a $(\mathcal{B}_2, \Sigma'_L)$-logical structure $\mathcal{S}_{\leq'}$ such that $\Sigma'_L = \{p_{ab}, p_1, p_0, p_a, p_b, p_{bc}, p_c\}$. The definition of the predicates in $\mathcal{S}_{\leq'}$ is given as above. For example, $p_a(w) = 1$ iff $w$ contains only $a$, only $b$, only $c$ or $b$ and $c$.

**Remark 2.2** The process of associating $\mathcal{S}_{\leq'}$ to $\mathcal{S}$ is reversible. Having $\mathcal{S}_{\leq'}$ we can obtain $\mathcal{S}$. To this, we just have to remark that $p^{\mathcal{S}}(\vec{a}) = b \in B - \{0'\}$ if

$$p_b^{\mathcal{S}_{\leq'}}(\vec{a}) = 1 \text{ and } p_x^{\mathcal{S}_{\leq'}}(\vec{a}) = 0, \text{ for all } x \succ' b,$$

for any $\vec{a} \in S_w$, where $w$ is the type of $p$. Moreover, $p^{\mathcal{S}}(\vec{a}) = 0'$ if $p_b^{\mathcal{S}_{\leq'}}(\vec{a}) = 0$, for all $b \succ' 0'$.

We emphasize that the process above is reversible because $\mathcal{S}_{\leq'}$ satisfies the following properties:

- for any $x, y \in B - \{0'\}$ with $x \geq' y$, if $p_x^{\mathcal{S}_{\leq'}}([\vec{a}]_\rho) = 1$, for some $p \in \Sigma_{L,w}$ and $\vec{a} \in S_w$, then $p_y^{\mathcal{S}_{\leq'}}([\vec{a}]_\rho) = 1$;

- for any $x, y_1, y_2 \in B - \{0'\}$ with $x \prec' y_1$ and $x \prec' y_2$, and any predicate $p \in \Sigma_{L,w}$, there exists no $\vec{a} \in S_w$ such that $p_{y_1}^{\mathcal{S}_{\leq'}}([\vec{a}]_\rho) = 1$ and $p_{y_2}^{\mathcal{S}_{\leq'}}([\vec{a}]_\rho) = 1$.

Unfortunately, the translation result does not work for any relation $\leq'$. This partial order must satisfy some constraints with respect to the equivalence relation used for abstraction.

**Definition 2.4** Let $\mathcal{S}$ and $\mathcal{S}_{\leq'}$ be as above and $\rho$ an equivalence relation on $S$. We say that $\leq'$ is *appropriate for* $\rho$ if:

- for any $x, y_1, y_2 \in B$ with $x \prec' y_1$ and $x \prec' y_2$, any predicate $p \in \Sigma_{L,w}$, and any $\vec{a} \in S_w$, $[\vec{a}]_\rho$ does not contain distinct elements $\vec{a_1}$ and $\vec{a_2}$ such that $p_{y_1}^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ and $p_{y_2}^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 1$.

The restriction on $\leq'$ will be the appropriateness for the equivalence considered for abstraction. We emphasize that this restriction still permits us to find a partial order $\leq'$, no matter what equivalence we choose for abstraction (we can always choose a linear order). Therefore, independently of the equivalence $\rho$ used for abstraction, there exist multi-valued $\alpha$-abstractions of logical structures by $\rho$ that can be obtained by using 2-valued abstractions.

**Example 2.6** Let $\mathcal{S}$ be the logical structure from Example 2.5. We consider an equivalence relation $\rho$ on $S_k$ such that $a^m b^n c^p \,\rho\, a^{m'} b^{n'} c^{p'}$ iff:

$(m = n = p = m' = n' = p' = 0)$ or
$(p = p' = 0 \wedge (m = m' = 0 \vee n = n' = 0 \vee m = n' = 0 \vee m' = n = 0))$ or
$(m > 0 \wedge n > 0 \wedge m' > 0 \wedge n' > 0)$ or
$(p > 0 \wedge p' > 0 \wedge (m > 0 \Rightarrow n = 0) \wedge (n > 0 \Rightarrow m = 0) \wedge (m' > 0 \Rightarrow n' = 0)$
$\wedge (n' > 0 \Rightarrow m' = 0))$.

Notice that $\leq'$ is appropriate for $\rho$ because we do not have equivalence classes containing the value 1 for $p_b$ and $p_{bc}$ or the value 1 for any two predicates from the set $\{p_{ab}, p_0, p_a\}$.

Since in the multi-valued abstractions, the predicates are defined according to the same interpretation policy we will consider $\mathcal{P}$-abstractions of $\mathcal{S}_{\leq'}$ for which $\mathcal{P}$ contains some predicate $p_b$ only if it contains all possible $p_b$ with $p \in \Sigma_L$. We will denote by $B_\mathcal{P}$ the set of truth values $b$ for which the predicates $p_b$ are in $\mathcal{P}$.

As it was the case with the partial order $\leq'$, we have some restrictions even for the set $\mathcal{P}$ of predicates.

**Definition 2.5** Let $\mathcal{S}_{\leq'}$ be as above, $\rho$ an equivalence relation on $S$, and $\mathcal{P} \subseteq \Sigma'_L$ such that $p_b \in \mathcal{P}$ iff $(\forall p \in \Sigma_L)(p_b \in \mathcal{P})$, for any $b \in B - \{0'\}$. We say that $\mathcal{P}$ is *appropriate for* $\rho$ if:

- for any $x, y \in B$ with $x \succ' y$, $x \notin B_{\mathcal{P}}$ and $y \in B_{\mathcal{P}}$, any predicate $p \in \Sigma_{L,w}$ and any $\vec{a} \in S_w$, we have the following:

$$(\exists \vec{a_1} \in [\vec{a}]_\rho)(p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1) \Rightarrow (\forall \vec{a_1} \in [\vec{a}]_\rho)(p_y^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1)$$

**Example 2.7** Let $\mathcal{S}$ and $\rho$ be as in Example 2.6. Moreover, let $\mathcal{P} = \{p_{ab}, p_1, p_b, p_{bc}\}$ be a subset of $\Sigma'_L$. We can easily see that $\mathcal{P}$ is appropriate for $\rho$ because the equivalence class containing elements for which $p_c$ is 1 has the property that $p_{bc}$ is 1 for all its elements.

Now, we prove that the $\mathcal{P}$-abstraction of $\mathcal{S}_{\leq'}$ by some equivalence $\rho$ is a logical structure from which we can still extract a multi-valued logical structure as in Remark 2.2, whenever $\leq'$ and $\mathcal{P}$ are appropriate for $\rho$.

**Theorem 2.7** Let $\mathcal{S}_{\leq'}$, $\rho$ and $\mathcal{P}$ be as above. If $\leq'$ and $\mathcal{P}$ are appropriate for $\rho$ then the $\mathcal{P}$-abstraction $\mathcal{S}'_{\leq'}$ of $\mathcal{S}_{\leq'}$ by $\rho$ satisfies the following properties:

1. for any $x, y \in B - \{0'\}$ with $x \geq' y$, any predicate $p \in \Sigma_{L,w}$ and any $\vec{a} \in S_w$:
$$p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1 \;\Rightarrow\; p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1;$$
2. for any $x, y_1, y_2 \in B - \{0'\}$ with $x \prec' y_1$ and $x \prec' y_2$, and any predicate $p \in \Sigma_{L,w}$, there is no $\vec{a} \in S_w$ such that:
$$p_{y_1}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = p_{y_2}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1.$$

**Proof** For (1), it is sufficient to prove that

$$p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1 \;\Rightarrow\; p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1,$$

for any $x, y \in B - \{0'\}$ with $x \succ' y$, any predicate $p \in \Sigma_{L,w}$, and any $\vec{a} \in S_w$. We consider the following cases:

- $x, y \notin B_{\mathcal{P}}$. By the definition of $\mathcal{S}'_{\leq'}$, $p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$ iff there exists $\vec{a_1} \in [\vec{a}]_\rho$ such that $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ and $p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$ iff there exists $\vec{a_2} \in [\vec{a}]_\rho$ such that $p_y^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 1$, for any $\vec{a} \in S_w$, where $w$ is the type of $p$. Since $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ implies $p_y^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ we also have that $p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$ implies $p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$;

- $x \in B_{\mathcal{P}}$ and $y \notin B_{\mathcal{P}}$. Suppose that $p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$. This implies $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and thus, $p_x^{\mathcal{S}_{\leq'}}(\vec{a}) = 1$. By the definition of $\mathcal{S}_{\leq'}$ we further obtain $p_y^{\mathcal{S}_{\leq'}}(\vec{a}) = 1$, which implies $p_y^{\mathcal{S}_{\leq'}}(\vec{a}) = 1$, by the fact that $y \notin B_{\mathcal{P}}$;

- $x \notin B_{\mathcal{P}}$ and $y \in B_{\mathcal{P}}$. Suppose that $p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$. This implies that there exists $\vec{a_1} \in [\vec{a}]_\rho$ such that $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ which, by the fact that $\mathcal{P}$ is appropriate for $\rho$, implies $p_y^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 1$, for all $\vec{a_2} \in [\vec{a}]_\rho$. Consequently, $p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$;

- $x, y \in B_{\mathcal{P}}$. Suppose that $p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$. This implies $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and thus, by the definition of $\mathcal{S}_{\leq'}$, $p_y^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$. Therefore, $p_y^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$.

To prove (2) we consider again several cases:

- $y_1, y_2 \in B_{\mathcal{P}}$. If we suppose that there exists some equivalence class $[\vec{a}]_\rho$ such that $p_{y_1}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = p_{y_2}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$, then $p_{y_1}^{\mathcal{S}_{\leq'}}(\vec{a_1}) = p_{y_2}^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$. However, this can not hold because $y_1$ and $y_2$ are incomparable with respect to $\leq'$;

- $y_1 \in B_{\mathcal{P}}$ and $y_2 \notin B_{\mathcal{P}}$. Suppose that there exists some equivalence class $[\vec{a}]_\rho$ such that $p_{y_1}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = p_{y_2}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$. Then, $p_{y_1}^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and $p_{y_2}^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 1$, for some $\vec{a_2} \in [\vec{a}]_\rho$. Again, we can use the fact that $y_1$ and $y_2$ are incomparable with respect to $\leq'$ to obtain that this contradicts the definition of $\mathcal{S}_{\leq'}$;

- $y_1 \notin B_{\mathcal{P}}$ $y_2 \in B_{\mathcal{P}}$. This is similar to the previous case;

- $y_1, y_2 \notin B_{\mathcal{P}}$. In order to have some equivalence class $[\vec{a}]_\rho$ with $p_{y_1}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = p_{y_2}^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$, there should exist $\vec{a_1}, \vec{a_2} \in [\vec{a}]_\rho$ such that $p_{y_1}^{\mathcal{S}_{\leq'}}(\vec{a_1}) = p_{y_2}^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 1$. However, this contradicts the fact that $\leq'$ is appropriate for $\rho$. □

The relationships between the logical structures we have build until now are depicted in Figure 2.2: $\mathcal{S}_{\leq'}$ is obtained from $\mathcal{S}$ by using the partial order $\leq'$, $\mathcal{S}'_{\leq'}$ is a $\mathcal{P}$-abstraction of $\mathcal{S}_{\leq'}$ by some equivalence $\rho$, and $\mathcal{S}'$ is obtained from $\mathcal{S}'_{\leq'}$ by using the partial order $\leq'$. We will now prove the dotted arrow

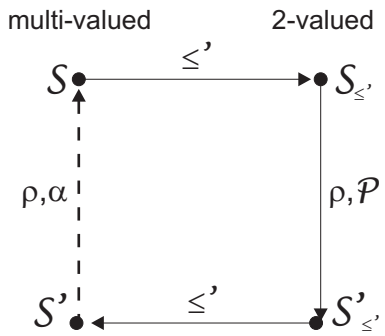


Figure 2.2: Relationships between multi-valued and 2-valued logical structures and abstractions.

by identifying the interpretation policy $\alpha$ that corresponds to $\mathcal{P}$ and $\leq'$, such that $\mathcal{S}'$ is an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$.

**Theorem 2.8** Let $\mathcal{S} = (S, \Sigma_L^S)$ be a $K$-kinded $(\mathcal{B}, \Sigma_L)$-logical structure, $\leq'$ a partial order on $B$, $\rho$ an equivalence relation on $S$, and $\mathcal{P} \subseteq \Sigma_L$ such that $\leq'$ and $\mathcal{P}$ are appropriate for $\rho$. If $\mathcal{S}'_{\leq'}$ is a $\mathcal{P}$-abstraction of $\mathcal{S}_{\leq'}$ by $\rho$ then $\mathcal{S}'$ is an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$, for some interpretation policy $\alpha$ which satisfies the following properties:

1. if $(\forall b \succ' 0')(b \notin B_{\mathcal{P}})$ then $\alpha(0') = \forall$;
2. if $(\exists b \succ' 0')(b \in B_{\mathcal{P}})$ then $\alpha(0') = \exists^T$, where $0' \in T$ and $\{x | x \geq' b\} \subseteq T$, for any $b \succ' 0'$ with $b \in B_{\mathcal{P}}$;
3. if $b \notin B_{\mathcal{P}}$ and $(\forall z \succ' b)(z \notin B_{\mathcal{P}})$ then $\alpha(b) = \exists^T$, where $T = \{x | c \leq' x \leq' b\}$ and $c$ is given by

$$c = \begin{cases} \text{the least element of } B \text{ such that } c \leq' b \text{ and } c \in B_{\mathcal{P}}, \text{if it exists;} \\ 0', \text{otherwise;} \end{cases}$$

4. if $b \notin B_{\mathcal{P}}$ and $(\exists z \succ' b)(z \in B_{\mathcal{P}})$ then $\alpha(b) = \exists_a^T$, where $T = \{x | x \geq' c\}$ and $c$ is given by

$$c = \begin{cases} \text{the least element of } B \text{ such that } c \leq' b \text{ and } c \in B_{\mathcal{P}}, \text{if it exists;} \\ 0', \text{otherwise;} \end{cases}$$

5. if $b \in B_{\mathcal{P}}$ and $(\forall z \succ' b)(z \notin B_{\mathcal{P}})$ then $\alpha(b) = \forall$;
6. if $b \in B_{\mathcal{P}}$ and $(\exists z \succ' b)(z \in B_{\mathcal{P}})$ then $\alpha(b) = \exists^T$, where $b \in T$ and $\{x | x \geq' z\} \subseteq T$, for any $z \succ' b$ with $z \in B_{\mathcal{P}}$;

7. if $b \in B_{\mathcal{P}}$ and $(\forall z)(z \not\succ' b)$ then $\alpha(b) = \forall$;

8. if $b \notin B_{\mathcal{P}}$ and $(\forall z)(z \not\succ' b)$ then $\alpha(b) = \exists^T$, where $T = \{x | x \geq' c\}$ and $c$ is given by

$$c = \begin{cases} \text{the least element of } B \text{ such that } c \leq' b \text{ and } c \in B_{\mathcal{P}}, \text{if it exists;} \\ 0', \text{otherwise;} \end{cases}$$

**Proof** Let $p \in \Sigma_{L,w}$ be a predicate and $\vec{a} \in S_w$. To prove the first and the second part we proceed as follows:

- if $p^{\mathcal{S}'}([\vec{a}]_\rho) = 0'$ then $p_b^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 0$, for all $b \succ' 0'$. Since $b \notin B_{\mathcal{P}}$, for any $b \succ' 0'$, we have that $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 0$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and $b \succ' 0'$. Thus, $p^{\mathcal{S}}(\vec{a_1}) = 0'$, for all $\vec{a_1} \in [\vec{a}]_\rho$, which implies $\alpha(0') = \forall$. This proves (1);

- if $p^{\mathcal{S}'}([\vec{a}]_\rho) = 0'$ then $p_b^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 0$, for all $b \succ' 0'$. Two cases are in order:

  - if $b \notin B_{\mathcal{P}}$ then $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 0$, for all $\vec{a_1} \in [\vec{a}]_\rho$;
  - if $b \in B_{\mathcal{P}}$ then $p_b^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 0$, for some $\vec{a_2} \in [\vec{a}]_\rho$.

  As $\leq'$ is appropriate for $\rho$, there exists a unique $b \succ' 0'$ such that $p_b(\vec{x}) = 1$, for some $\vec{x} \in [\vec{a}]_\rho$. Consequently, there should exist $\vec{a_3} \in [\vec{a}]_\rho$ such that $p_b^{\mathcal{S}_{\leq'}}(\vec{a_3}) = 0$, for all $b \succ' 0'$, which implies $p^{\mathcal{S}}(\vec{a_3}) = 0'$. Moreover, for any $\vec{a_1} \in [\vec{a}]_\rho$, we can have $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ only if $b \in B_{\mathcal{P}}$ and, consequently, the values different from $0'$ of $p^{\mathcal{S}}(\vec{a_1})$ are greater than or equal to some $b \succ' 0'$ with $b \in B_{\mathcal{P}}$. This proves (2).

Now, let $b \in B - \{0'\}$ such that $p^{\mathcal{S}'}([\vec{a}]_\rho) = b$. First, we consider the case when there exists $z \in B$ with $z \succ' b$. We obtain that:

$$p_b^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1 \text{ and } p_x^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 0, \text{ for all } x \succ' b.$$

The following cases are to be discussed:

- if $b \notin B_{\mathcal{P}}$ and $(\forall z \succ' b)(z \notin B_{\mathcal{P}})$ then there exists $\vec{a_1} \in [\vec{a}]_\rho$ such that $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ and $p_x^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 0$, for all $\vec{a_2} \in [\vec{a}]_\rho$ and $x \succ' b$. This clearly implies $p^{\mathcal{S}}(\vec{a_1}) = b$ and $p^{\mathcal{S}}(\vec{a_2}) \leq' b$, for all $\vec{a_2} \in [\vec{a}]_\rho$.

  Assume that there exists the least element $c \in B$ such that $c \leq' b$ and $c \in B_{\mathcal{P}}$. Suppose by contradiction that there exists $\vec{a_3} \in [\vec{a}]_\rho$ such that $p^{\mathcal{S}}(\vec{a_3}) <' c$. We obtain $p_c^{\mathcal{S}_{\leq'}}(\vec{a_3}) = 0$ which, by $c \in B_{\mathcal{P}}$,

implies $p_c^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 0$. Further, by the construction of $\mathcal{S}'_{\leq'}$, we get $p_b^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 0$, which contradicts the hypothesis.

If no such $c$ exists, then we can not impose any more restrictions on the value of $p^{\mathcal{S}}(\vec{a_1})$ with $\vec{a_1} \in [\vec{a}]_\rho$. This proves (3);

- if $b \notin B_{\mathcal{P}}$ and $(\exists z \succ' b)(z \in B_{\mathcal{P}})$ then:
  - there exists $\vec{a_1} \in [\vec{a}]_\rho$ such that $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$;
  - there exists $\vec{a_x} \in [\vec{a}]_\rho$ such that $p_x^{\mathcal{S}_{\leq'}}(\vec{a_x}) = 0$, for all $x \succ' b$ with $x \in B_{\mathcal{P}}$;
  - $p_x^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 0$, for all $\vec{a_2} \in [\vec{a}]_\rho$ and $x \succ' b$ with $x \notin B_{\mathcal{P}}$.

  By the construction of $\mathcal{S}_{\leq'}$, we obtain $p^{\mathcal{S}}(\vec{a_1}) \geq' b$ and $p^{\mathcal{S}}(\vec{a_x}) \leq' b$. The fact that $p^{\mathcal{S}}(\vec{a_3}) \geq' c$, for all $\vec{a_3} \in [\vec{a}]_\rho$, is obtained exactly as in the previous case. This proves (4);

- if $b \in B_{\mathcal{P}}$ and $(\forall z \succ' b)(z \notin B_{\mathcal{P}})$ then $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ and $p_x^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 0$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and $x \succ' b$. Obviously, we get $p^{\mathcal{S}}(\vec{a_1}) = b$, for all $\vec{a_1} \in [\vec{a}]_\rho$ and therefore, $\alpha(b) = \forall$. This proves (5);

- if $b \in B_{\mathcal{P}}$ and $(\exists z \succ' b)(z \in B_{\mathcal{P}})$ then:
  - $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$;
  - there exists $\vec{a_x} \in [\vec{a}]_\rho$ such that $p_x^{\mathcal{S}_{\leq'}}(\vec{a_x}) = 0$, for all $x \succ' b$ with $x \in B_{\mathcal{P}}$;
  - $p_x^{\mathcal{S}_{\leq'}}(\vec{a_2}) = 0$, for all $\vec{a_2} \in [\vec{a}]_\rho$ and $x \succ' b$ with $x \notin B_{\mathcal{P}}$.

  Clearly, the first property implies $p^{\mathcal{S}}(\vec{a_1}) \geq' b$, for all $\vec{a_1} \in [\vec{a}]_\rho$, while the third implies that there exists no $\vec{a_2} \in [\vec{a}]_\rho$ such that $p^{\mathcal{S}}(\vec{a_2}) \geq' x$, where $x \succ' b$ and $x \notin B_{\mathcal{P}}$. This proves (6).

Finally, let $b \in B - \{0'\}$ such that $(\forall z)(z \not\succ' b)$ and $p^{\mathcal{S}'}([\vec{a}]_\rho) = b$. We obtain that $p_b^{\mathcal{S}'_{\leq'}}([\vec{a}]_\rho) = 1$ and we consider two cases:

- if $b \in B_{\mathcal{P}}$ then $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$, for all $\vec{a_1} \in [\vec{a}]_\rho$. This implies $p^{\mathcal{S}}(\vec{a_1}) = b$, for all $\vec{a_1} \in [\vec{a}]_\rho$, and therefore, $\alpha(b) = \forall$. This proves (7);

- if $b \notin B_{\mathcal{P}}$ then there exists $\vec{a_1} \in [\vec{a}]_\rho$ such that $p_b^{\mathcal{S}_{\leq'}}(\vec{a_1}) = 1$ which implies $p^{\mathcal{S}}(\vec{a_1}) = b$. Moreover, as in case (3), we can obtain $p^{\mathcal{S}}(\vec{a_2}) \geq' c$, for all $\vec{a_2} \in [\vec{a}]_\rho$. This proves (8). □

**Example 2.8** Let $\mathcal{S}$, $\mathcal{S}_{\leq'}$, $\rho$ and $\mathcal{P}$ be as in Example 2.7 and $\mathcal{S}'_{\leq'}$ the $\mathcal{P}$-abstraction of $\mathcal{S}_{\leq'}$ by $\rho$. By Theorem 2.8, we can find the following interpretation policy $\alpha$ such that $\mathcal{S}'$, the $(\mathcal{B}, \Sigma_L)$-logical structure corresponding to $\mathcal{S}'_{\leq'}$, is an $\alpha$-abstraction of $\mathcal{S}$ by $\rho$:

| Truth values | $ac$ | $ab$ | 1 | 0 | $a$ | $b$ | $bc$ | $c$ |
|---|---|---|---|---|---|---|---|---|
| The policy $\alpha$ | $\exists^{\{ab,1,ac\}}$ | $\exists^{\{ab,1\}}$ | $\forall$ | $\exists$ | $\exists_a$ | $\forall$ | $\forall$ | $\exists^{\{bc,c\}}$ |

## 2.3 Temporal Logic

### 2.3.1 A Case Study: Abstractions of Protection Systems

Before we describe multi-valued abstractions of Kripke structures, we present a case study of using abstraction in the context of access control models [44] (the abstraction technique used is *the simulation relation* [94]).

Access control is one of the facets of the implementation of security policies. In access control models, the security policy is implemented by an assignment of access rights to the objects composing the system and by the rules allowing the creation and/or destruction of new objects and the modification of their access rights.

A powerful model of access control systems is the *access matrix* model [63]. In this model, the *protection state* of the system is characterized by the set of access rights that different entities (*subjects* or *objects*) have over other entities and by the set of *commands* which may change this state, by creating/destroying subjects or objects or by adding/removing rights. The expressive power of this model is sufficiently large to include other models like take-grant systems [78], SPM systems [107], ESPM systems [1], TAM systems [106] etc.

The basic decision problem in an access matrix model is the *safety problem*: given two entities $A$ and $B$ and a right $R$, decide whether the system can evolve into a state in which $A$ has right $R$ over $B$. Very early, it was shown that this problem is undecidable [63] and remains like that even for systems without subject/object destruction [62]. Consequently, a number of restrictions have been proposed [63, 78, 106] for which the safety problem is decidable.

We propose two notions of simulation between protection systems and define a class of access control models that are simulated by access control

models with a finite number of objects for which the safety problem is decidable. Then, we show that several classes of protection systems from the literature fall into this class, notably the *take-grant systems* and the *monotonic typed access matrix systems with an acyclic creation graph*. By this we also unify and clarify the proof of decidability of the safety problem for these classes of protection systems.

### Protection Systems

We use protection systems modeled as in [63]. Here, the protection state of a system is modeled by an access matrix with a row for each subject and a column for each object. The cells hold the rights that subjects have on objects.

A protection system is defined over a finite set of generic rights and contains commands that specify how the protection state can be changed. The commands are formed of a conditional part which tests for the presence of rights in some cells of the access matrix and an operational part which specifies the changes made on the protection state. The changes are specified using primitive operations for subject/object creation and destruction and for entering/removing rights.

**Definition 2.6** A *protection scheme* is a tuple $\mathcal{S} = (R, C)$, where $R$ is a finite set of *rights* and $C$ is a finite set of *commands* of the following form:

$$
\begin{array}{l}
\text{command } c(x_1, x_2, \cdots, x_n) \\
\quad \text{if} \quad r_1 \quad \text{in} \quad [x_{s_1}, x_{o_1}] \\
\qquad \cdots \\
\qquad r_k \quad \text{in} \quad [x_{s_k}, x_{o_k}] \\
\quad \text{then} \\
\qquad op_1 \\
\qquad \cdots \\
\qquad op_m
\end{array}
$$

Above, $c$ is a name, $x_1, x_2, \cdots, x_n$ are formal parameters and each $op_i$ is one of the following *primitive operations*: enter $r$ into $[x_s, x_o]$, delete $r$ from $[x_s, x_o]$, create subject $x_s$, create object $x_o$, destroy subject $x_s$, and destroy object $x_o$. Also, $r, r_1, \cdots r_k$ are rights from $R$ and $s$, $s_1$,$\cdots$, $s_k$, $o$, $o_1$,$\cdots$, $o_k$ are integers between 1 and $n$.

We will call a command *mono-operational* if it contains only one primitive operation and *monotonic* if it does not contain "destroy subject", "destroy object" and "delete" operations.

**Definition 2.7** A *configuration* over $R$ is a tuple $Q = (S, O, P)$, where $S$ is the set of *subjects*, $O$ the set of *objects*, $S \subseteq O$, and $P : S \times O \to \mathcal{P}(R)$ is the *access matrix*. We will denote by $Cf(R)$ the set of configurations over $R$.

As we can see, all subjects are also objects. This is a very natural assumption since, for example, processes in a computer system may be accessed by, or may access other processes. The objects from $O - S$ will be called *pure objects*.

**Definition 2.8** A *protection system* is a tuple $\psi = (R, C, Q_0)$, where $(R, C)$ is a protection scheme and $Q_0$ a configuration over $R$, called the *initial configuration*. A protection system is *mono-operational* (*monotonic*) if all commands in $C$ are mono-operational (monotonic).

We will call the subjects (objects) from $S_0$ ($O_0$) *initial subjects (objects)*.

The six primitive operations mean exactly what their name imply (for details the reader is referred [63]). We will denote by $\Rightarrow_{op}$ the application of a primitive operation *op* in some configuration.

**Definition 2.9** Let $\psi = (R, C, Q_0)$ be a protection system, $Q$ and $Q'$ two configurations over $R$ and $c(x_1, \cdots x_n) \in C$ a command like in Definition 2.6. We say that $Q'$ *is obtained from* $Q$ *in* $\psi$*, applying* $c$ *with the actual arguments* $o_1, \cdots, o_n$, denoted by $Q \to_{\psi}^{c(o_1,\cdots,o_n)} Q'$, if:

- $r_i \in P(o_{s_i}, o_{o_i})$, for all $1 \leq i \leq k$;
- there exist configurations $Q_1, \cdots, Q_m$ such that $Q \Rightarrow_{op'_1} Q_1 \Rightarrow_{op'_2} \cdots \Rightarrow_{op'_m} Q_m$ and $Q_m = Q'$ ($op'_i$ is the primitive operation obtained after substituting $x_1, \cdots, x_n$ with $o_1, \cdots, o_n$).

When the command $c$ and the actual arguments $o_1, \cdots, o_n$ are understood from the context, we will write only $Q \to_{\psi} Q'$. We consider also $\to_{\psi}^*$, the reflexive and transitive closure of $\to_{\psi}$. We say that a configuration $Q$ over $R$ is *reachable in* $\psi$ if $Q_0 \to_{\psi}^* Q$.

For protection systems like above, we consider the following safety problem: given $s$ an initial subject, $o$ an initial object and a right $r$, decide if a state in which $s$ has right $r$ over $o$ is reachable. This is a less general safety problem than the one in [63], but it is more natural and used more frequently in the literature [1, 78, 107, 106].

**Definition 2.10** Let $\psi = (R, C, Q_0 = (S_0, O_0, P_0))$ be a protection system, $s \in S_0$, $o \in O_0$ and $r \in R$. A configuration $Q = (S, O, P)$ over $R$ is called *leaky for* $(s, o, r)$ if $s \in S$, $o \in O$ and $r \in P(s, o)$.

We say that $\psi$ is *leaky for* $(s, o, r)$ if there exists a reachable configuration leaky for $(s, o, r)$. Otherwise, $\psi$ is called *safe for* $(s, o, r)$, denoted by $\psi \triangleleft (s, o, r)$.

Now we can define the safety problem as follows:

**Safety problem** (SP)
**Instance:** A protection system $\psi$, $s \in S_0$, $o \in O_0$, $r \in R$;
**Question:** is $\psi$ safe for $(s, o, r)$?

### Simulations

The problem to decide if a protection system is safe was shown to be undecidable in [63], by designing a protection system that simulates a Turing machine. The most important source of undecidability is the creation of objects, which makes the system infinite-state. Hence, techniques to reduce the state space of the system are well suited. In this paper we will refer to an abstraction technique, namely the *simulation relation* [94].

**Definition 2.11** Let $\psi_1 = (R_1, C_1, Q_0^1 = (S_0^1, O_0^1, P_0^1))$ and $\psi_2 = (R_2, C_2, Q_0^2 = (S_0^2, O_0^2, P_0^2))$ be two protection systems. Also, let $\rho_o \subseteq O_0^1 \times O_0^2$ and $\rho_r \subseteq R_1 \times R_2$ be two relations. For any $Q_1 = (S_1, O_1, P_1) \in Cf(R_1)$ and $Q_2 = (S_2, O_2, P_2) \in Cf(R_2)$, we say that $Q_2$ *simulates* $Q_1$ *w.r.t.* $\rho_o$ *and* $\rho_r$, denoted by $Q_1 \prec_{\rho_o, \rho_r} Q_2$, if:

1. $\rho_o(S_1 \cap S_0^1) \subseteq S_2 \cap S_0^2$;
2. $\rho_o(O_1 \cap O_0^1) \subseteq O_2 \cap O_0^2$;
3. for any $s \in S_1 \cap S_0^1$, $o \in O_1 \cap O_0^1$ and $r \in R_1$, if $r \in P_1(s, o)$ then there exist $s' \in \rho_o(s)$, $o' \in \rho_o(o)$ and $r' \in \rho_r(r)$ such that $r' \in P_2(s', o')$.

Above, $\rho(s) = \{s' | \rho(s, s')\}$, for any relation $\rho$.

The relations $\rho_o$ and $\rho_r$ are used to relate the "access powers" of subjects from two different protection systems. For example, having right $r$ over an object $o$ in the first system is considered to be the same as having a right $r' \in \rho_r(r)$ over an object $o' \in \rho_o(o)$ in the second system. In this context, a configuration $Q_2$ from $\psi_2$ simulates a configuration $Q_1$ from $\psi_1$, if every initial subject from $Q_2$ has at least the same "access power" as the initial subject from $Q_1$ to which is related by $\rho_o$.

The simulation relation we define next is more general than the one in [94] because one transition step in the first system can be simulated by zero, one, or more transition steps in the second system.

**Definition 2.12** Let $\psi_1 = (R_1, C_1, Q_0^1)$ and $\psi_2 = (R_2, C_2, Q_0^2)$ be two protection systems, and $\rho_o$, $\rho_r$ relations like above. We say that $H \subseteq Cf(R_1) \times Cf(R_2)$ is a *simulation relation from $\psi_1$ to $\psi_2$ w.r.t. $\rho_o$ and $\rho_r$* if for any $Q_1 \in Cf(R_1)$ and $Q_2 \in Cf(R_2)$, $H(Q_1, Q_2)$ implies that:

1. $Q_1 \prec_{\rho_o,\rho_r} Q_2$;
2. for any $Q_1' \in Cf(R_1)$ such that $Q_1 \rightarrow_{\psi_1} Q_1'$ there exists $Q_2' \in Cf(R_2)$ such that $Q_2 \rightarrow_{\psi_2}^* Q_2'$ and $H(Q_1', Q_2')$.

**Definition 2.13** Let $\psi_1 = (R_1, C_1, Q_0^1)$ and $\psi_2 = (R_2, C_2, Q_0^2)$ be two protection systems, and $\rho_o$, $\rho_r$ relations like above. We say that $\psi_2$ *simulates* $\psi_1$ *w.r.t. $\rho_o$ and $\rho_r$*, denoted by $\psi_1 \prec_{\rho_o,\rho_r} \psi_2$, if there exists a simulation relation $H$ from $\psi_1$ to $\psi_2$ w.r.t. $\rho_o$ and $\rho_r$ such that $H(Q_0^1, Q_0^2)$. We write $\psi_1 \prec \psi_2$ if there exist $\rho_o$ and $\rho_r$ like above such that $\psi_1 \prec_{\rho_o,\rho_r} \psi_2$.

The usefulness of the simulation relation is proved by the next theorem. We will show that solving some instances of SP in a protection system that simulates another may lead to solving an instance of SP in the initial system.

**Theorem 2.9** Let $\psi_1 = (R_1, C_1, Q_0 = (S_0, O_0, P_0))$ and $\psi_2 = (R_2, C_2, Q_0' = (S_0', O_0', P_0'))$ be two protection systems, and $\rho_o$, $\rho_r$ two relations like above. If $\psi_1 \prec_{\rho_o,\rho_r} \psi_2$, then:

$$\big[(\forall s' \in \rho_o(s))(\forall o' \in \rho_o(o))(\forall r' \in \rho_r(r))(\psi_2 \triangleleft (s', o', r'))\big] \Rightarrow \big[\psi_1 \triangleleft (s, o, r)\big],$$

for any $s \in S_0$, $o \in O_0$ and $r \in R_1$.

**Proof** Suppose by contradiction that $\psi_1$ is not safe for $(s, o, r)$. Then, there exists the following computation in $\psi_1$:

$$Q_0 \rightarrow_{\psi_1} Q_1 \rightarrow_{\psi_1} \cdots \rightarrow_{\psi_1} Q_l,$$

such that $Q_l$ is leaky for $(s, o, r)$.

$\psi_1 \prec_{\rho_o,\rho_r} \psi_2$ implies that there exists a simulation relation $H$ from $\psi_1$ to $\psi_2$ such that $H(Q_0, Q_0')$. Hence, we have in $\psi_2$ the following computation:

$$Q_0' \rightarrow_{\psi_2}^* Q_1' \rightarrow_{\psi_2}^* \cdots \rightarrow_{\psi_2}^* Q_l',$$

such that $H(Q_i, Q_i')$ for all $0 \leq i \leq l$.

Consequently, $Q_l \prec_{\rho_o,\rho_r} Q_l'$ and, since $r \in P_l(s, o)$, we obtain that there exists $s'$ in $\rho_o(s)$, $o'$ in $\rho_o(o)$ and $r'$ in $\rho_r(r)$ such that $r' \in P_l'(s', o')$, where $P_l$ is the access matrix of $Q_l$ and $P_l'$ the access matrix of $Q_l'$. So, $Q_l'$ is leaky for $(s', o', r')$ and $\psi_2$ is not safe for $(s', o', r')$, contradicting the supposition made above. □

The result above implies that $\psi_2$ is a *weak-preserving abstraction* of $\psi_1$ in the sense that only positive answers to instances of SP in $\psi_2$ may lead to solving instances of SP in $\psi_1$.

We will now prove that, in some conditions, the existence of simulation relations in both senses may transform $\psi_2$ into a *strong-preserving abstraction* of $\psi_1$. This means that, also negative answers to instances of SP in $\psi_2$ are important for solving instances of SP in $\psi_1$.

We say that a relation $\rho \subseteq D_1 \times D_2$ is *injective* if $\rho(x_1) \cap \rho(x_2) = \emptyset$, for any $x_1, x_2 \in D_1$.

**Corollary 2.7** Let $\psi_1 = (R_1, C_1, Q_0 = (S_0, O_0, P_0))$ and $\psi_2 = (R_2, C_2, Q_0' = (S_0', O_0', P_0'))$ be two protection systems, and $\rho_o$, $\rho_r$ two relations like above.

If $\psi_1 \prec_{\rho_o,\rho_r} \psi_2$, $\psi_2 \prec_{\rho_o^{-1},\rho_r^{-1}} \psi_1$ and $\rho_o$, $\rho_r$ are injective then:

$$\big[(\forall s' \in \rho_o(s))(\forall o' \in \rho_o(o))(\forall r' \in \rho_r(r))(\psi_2 \triangleleft (s', o', r'))\big] \Leftrightarrow \big[\psi_1 \triangleleft (s, o, r)\big],$$

for any $s \in S_0$, $o \in O_0$ and $r \in R_1$.

**Proof** The result is immediate, applying Theorem 2.9 for $\psi_1 \prec_{\rho_o,\rho_r} \psi_2$ and $\psi_2 \prec_{\rho_o^{-1},\rho_r^{-1}} \psi_1$. □

Next, we exemplify the use of simulation relations in the analysis of protection systems. We will show that an weak-preserving abstraction of a protection system can be obtained by adding commands or by removing the non-monotonic primitive operations from all the commands. Then, for monotonic protection systems, we prove that an abstraction can be obtained by splitting each command into mono-operational commands.

**Example 2.9** Let $\psi = (R, C, Q_0 = (S_0, O_0, P_0))$ and $\psi' = (R, C', Q_0)$ be two protection systems such that $C \subseteq C'$.

We say that two configurations $Q = (S, O, P)$ and $Q' = (S', O', P')$ from $Cf(R)$ are *equal up to names*, denoted by $Q \approx Q'$, if:

- $O \cap O_0 = O' \cap O_0$;
- there exists a bijection $\phi : O \to O'$ such that:
  - $\phi(o) = o$, for every $o \in O \cap O_0$ ($\phi$ preserves initial objects);
  - $\phi(S) = S'$ ($\phi$ preserves subjects);
  - $r \in P(s, o) \Leftrightarrow r \in P'(\phi(s), \phi(o))$, for any $s \in S$, $o \in O$ and $r \in R$.

It can be easily proved that $\psi \prec_{id_{O_0},id_R} \psi'$, considering the simulation relation $H \subseteq Cf(R) \times Cf(R)$, given by $H(Q, Q')$ iff $Q \approx Q'$.

**Example 2.10** Let $\psi = (R, C, Q_0 = (S_0, O_0, P_0))$ be a protection system. We suppose w.l.o.g. that the commands in $C$ do not delete a right that they have just entered or destroy an object that they have just created. We will prove that $\psi$ is simulated by its *monotonic restriction*, i.e., by the system which acts like $\psi$ but does not destroy any object and does not delete any right.

Let $\psi_m = (R, C_m, Q_0)$, where $C_m$ is the set of commands obtained from the ones in $C$, by removing all non-monotonic primitive operations.

We can prove that $\psi \prec_{id_{O_0}, id_R} \psi_m$, considering the following relation $H$: given $Q = (S, O, P)$ and $Q' = (S', O', P')$ from $Cf(R)$, we have $H(Q, Q')$ if:

- $O \cap O_0 \subseteq O' \cap O_0$;
- there exists an injection $\phi : O \to O'$, such that:
  - $\phi(o) = o$, for every $o \in O \cap O_0$;
  - $\phi(S) \subseteq S'$;
  - $r \in P(s, o) \Rightarrow r \in P'(\phi(s), \phi(o))$, for any $s \in S$, $o \in O$ and $r \in R$.

**Example 2.11** Let $\psi = (R, C, Q_0)$ be a monotonic protection system. We can prove that $\psi$ is simulated by the mono-operational system that results by splitting all the commands from $C$ into mono-operational commands.

To this end, we will reuse the relation $H$ defined by $H(Q, Q')$ if $Q \approx Q'$ and prove that it is a simulation from $\psi$ to $\psi_{mo}$ w.r.t. $id_{O_0}$ and $id_R$.

### Quasi-bisimulations

We present another type of simulation relation between protection systems, that resembles to a bisimulation relation [98] but does not induce an equivalence relation over protection systems. That is why we will call this relation a quasi-bisimulation. It differs from the simulation relation presented earlier by the fact that initial subjects must have the same "access power" and we must be able to simulate one step from a protection system with a sequence of zero or more steps in the other system.

**Definition 2.14** Let $\psi_1 = (R_1, C_1, Q_0^1 = (S_0^1, O_0^1, P_0^1))$ and $\psi_2 = (R_2, C_2, Q_0^2 = (S_0^2, O_0^2, P_0^2))$ be two protection systems. Also, let $\rho_o \subseteq O_0^1 \times O_0^2$ and $\rho_r \subseteq R_1 \times R_2$ be two relations. For any $Q_1 = (S_1, O_1, P_1) \in Cf(R_1)$ and $Q_2 = (S_2, O_2, P_2) \in Cf(R_2)$, we say that $Q_2$ *is quasi-bisimilar to* $Q_1$ *w.r.t.* $\rho_o$ *and* $\rho_r$, denoted by $Q_1 \preceq_{\rho_o, \rho_r} Q_2$, if:

1. $\rho_o(S_1 \cap S_0^1) \subseteq S_2 \cap S_0^2$;

2. $\rho_o(O_1 \cap O_0^1) \subseteq O_2 \cap O_0^2$;

3. For any $s \in S_1 \cap S_0^1$, $o \in O_1 \cap O_0^1$ and $r \in R_1$, $r \in P_1(s,o)$ iff there exist $s' \in \rho_o(s)$, $o' \in \rho_o(o)$ and $r' \in \rho_r(r)$ such that $r' \in P_2(s',o')$.

**Definition 2.15** Let $\psi_1 = (R_1, C_1, Q_0^1)$ and $\psi_2 = (R_2, C_2, Q_0^2)$ be two protection systems, and $\rho_o$, $\rho_r$ relations like above. We say that $B \subseteq Cf(R_1) \times Cf(R_2)$ is a *quasi-bisimulation relation from $\psi_1$ to $\psi_2$ w.r.t. $\rho_o$ and $\rho_r$* if for any $Q_1 \in Cf(R_1)$ and $Q_2 \in Cf(R_2)$, $B(Q_1, Q_2)$ implies that:

1. $Q_1 \preceq_{\rho_o,\rho_r} Q_2$

2. for any $Q_1' \in Cf(R_1)$ such that $Q_1 \rightarrow_{\psi_1} Q_1'$ there exists $Q_2' \in Cf(R_2)$ such that $Q_2 \rightarrow_{\psi_2}^* Q_2'$ and $B(Q_1', Q_2')$.

3. for any $Q_2' \in Cf(R_2)$ such that $Q_2 \rightarrow_{\psi_2} Q_2'$ there exists $Q_1' \in Cf(R_1)$ such that $Q_1 \rightarrow_{\psi_1}^* Q_1'$ and $B(Q_1', Q_2')$.

**Definition 2.16** Let $\psi_1 = (R_1, C_1, Q_0^1)$ and $\psi_2 = (R_2, C_2, Q_0^2)$ be two protection systems, and $\rho_o$, $\rho_r$ relations like above. We say that $\psi_2$ *is quasi-bisimilar to $\psi_1$ w.r.t. $\rho_o$ and $\rho_r$*, denoted by $\psi_1 \preceq_{\rho_o,\rho_r} \psi_2$, if there exists a quasi-bisimulation relation $B$ from $\psi_1$ to $\psi_2$ w.r.t. $\rho_o$ and $\rho_r$, such that $B(Q_0^1, Q_0^2)$. We write $\psi_1 \preceq \psi_2$ if there exist $\rho_o$ and $\rho_r$ like above such that $\psi_1 \preceq_{\rho_o,\rho_r} \psi_2$.

Next, we will prove the usefulness of the quasi-bisimulations, showing that if we have two protection systems such that $\psi_1 \preceq \psi_2$, solving an instance of SP in $\psi_1$ is equivalent with solving one or more instances of SP in $\psi_2$. This could still be more efficient if the state space of $\psi_2$ is much smaller than the one of $\psi_1$.

**Theorem 2.10** Let $\psi_1 = (R_1, C_1, Q_0 = (S_0, O_0, P_0))$ and $\psi_2 = (R_2, C_2, Q_0' = (S_0', O_0', P_0'))$ be two protection systems, and $\rho_o$, $\rho_r$ two relations like above. If $\psi_1 \preceq_{\rho_o,\rho_r} \psi_2$, then:

$$\big[(\forall s' \in \rho_o(s))(\forall o' \in \rho_o(o))(\forall r' \in \rho_r(r))(\psi_2 \triangleleft (s', o', r'))\big] \Leftrightarrow \big[\psi_1 \triangleleft (s, o, r)\big],$$

for any $s \in S_0$, $o \in O_0$ and $r \in R_1$.

**Proof** Similar to the proof of Theorem 2.9. □

### A Class of Decidable Protection Systems

**Definition 2.17** A protection system $\psi = (R, C, Q_0)$ is called *a finite protection system* if the commands from $C$ do not contain "create" primitive operations.

It is well known [79] that the safety problem for finite protection systems is decidable.

**Definition 2.18** We define $Dec$ to be the class of protection systems that has the following properties:

- if $\psi$ is a finite protection system then $\psi \in Dec$;
- if $\psi' \in Dec$ and $\psi \prec_{\rho_o,\rho_r} \psi'$, $\psi' \prec_{\rho_o^{-1},\rho_r^{-1}} \psi$, for some $\rho_o$ and $\rho_r$ injective relations, then $\psi \in Dec$;
- if $\psi' \in Dec$ and $\psi \preceq \psi'$ then $\psi \in Dec$.

By Corollary 2.7 and Theorem 2.10, we obtain that the safety problem for the protection systems from $Dec$ is decidable.

We will show that it contains three other well-known classes of protection systems for which the safety problem is decidable: MTAM systems with acyclic creation graphs [106], mono-operational protection systems [63] and take-grant systems [78].

By showing that these three classes of protection systems are included in $Dec$, we unify their proof of decidability for the safety problem. We show that the safety problem is decidable because these protection systems are simulated by systems that have all the needed objects created even from the initial configuration and no commands that can create objects afterward.

**MTAM Systems with Acyclic Creation Graphs** In [106], the authors propose an extension of the a access matrix model, the typed access matrix model (TAM, for short), that assigns a type to each object of a configuration.

Formally, a TAM system is a tuple $\tau = (R, T, C, Q_0 = (S_0, O_0, t_0, P_0))$, where $R$ is a finite set of rights, $T$ a finite set of types, $C$ a finite set of typed commands and $Q_0$ the initial configuration.

Configurations are tuples $Q = (S, O, t, P)$, where $S$, $O$ and $P$ are as before, and $t : O \to T$ is a function that assigns a type to every object.

The typed commands differ from the commands of a protection system defined as in Sect. 2, by the fact that they test the type of each argument and the primitive operations used to create objects are: "create subject $s$ of type $t$" and "create object $o$ of type $t$".

$\tau$ is called a monotonic TAM system (MTAM, for short) if the commands from $C$ do not contain operations that delete rights or destroy objects.

We say that an MTAM system is in canonical form if the "create" commands (commands that contain at least one "create" primitive operation) are unconditional (the conditional part is empty). In [106] was proved that MTAM systems can always be considered to be in canonical form.

If $c$ is a typed command like in Fig. 2.3, $t_i$ is called a child type of $c$ if "create subject $x_i$ of type $t_i$" or "create object $x_i$ of type $t_i$" appears in $c$. Otherwise, $t_i$ is called a parent type of $c$.

The creation graph of a TAM system $\tau = (R, T, C, Q_0)$ is a directed graph with the set of vertexes $T$ and an edge from $t_1$ to $t_2$ if there exists a command $c \in C$ such that $t_1$ is a parent type of $c$ and $t_2$ a child type of $c$.

The safety problem SP can be defined analogously for TAM systems.

The main decidability result of [106] is:

**Theorem**
*SP for MTAM systems with acyclic creation graphs is decidable.*

A TAM system $\tau = (R, T, C, Q_0 = (S_0, O_0, t_0, P_0))$ can be described using a protection system $\psi_\tau = (R \cup T, C', Q_0' = (O_0, O_0, P_0'))$, where:

$$P_0'(s,o) = \begin{cases} P_0(s,o), & \text{if } s \neq o \text{ and } s \in S_0 \\ P_0(s,o) \cup \{t_0(s)\}, & \text{if } s = o \text{ and } s \in S_0 \\ \{t_0(s)\}, & \text{if } s = o \text{ and } s \in O_0 - S_0 \\ \emptyset, & \text{otherwise.} \end{cases}$$

and $C' = \{\gamma(c) | c \in C\}$, with $\gamma$ the transformation from Fig. 2.3 ($i_1, \cdots,$ $i_l$ are integers between 1 and n such that $x_{i_l}$ does not appear in a "create" operation).

The primitive operations of $\gamma(c)$ are obtained by copying the ones from $c$, excepting the case of a "create subject $s$ of type $t$" or "create object $s$ of type $t$" primitive operation, when we add in $\gamma(c)$ two operations: "create subject $s$" or "create object $s$" and "enter $t$ in [s,s]".

From now on, when we say TAM systems, we mean protection systems like $\psi_\tau$. Hence, an MTAM system is in canonical form if in the conditional part of every "create" command we test only rights from $T$.

In the following, we will obtain using quasi-bisimulations, the decidability of the safety problem for a class of protection systems more general than the MTAM systems with acyclic creation graphs.

If $\psi = (R, C, Q_0)$ is a protection system, denote by $\mathcal{T}_\psi(\mathcal{X})$ the set of terms

command $c(x_1 : t_1, x_2 : t_2, \cdots, x_n : t_n)$

if $r_1$ in $[x_{s_1}, x_{o_1}]$
$\cdots$
$r_k$ in $[x_{s_k}, x_{o_k}]$
then
$op_1$
$\cdots$
$op_m$

command $\gamma(c)(x_1, x_2, \cdots, x_n)$

if $t_{i_1}$ in $[x_{i_1}, x_{i_1}]$
$\cdots$
$t_{i_l}$ in $[x_{i_l}, x_{i_l}]$
$r_1$ in $[x_{s_1}, x_{o_1}]$
$\cdots$
$r_k$ in $[x_{s_k}, x_{o_k}]$
then
$op'_1$
$\cdots$
$op'_{m'}$

Figure 2.3: The transformation $\gamma$.

defined over the set of variables $\mathcal{X}$ and the signature $\Sigma_\psi$, where

$$\begin{aligned}\Sigma_\psi \;=\; & \{c_i \text{ of arity } n \mid c(x_1, \cdots, x_n) \in C \text{ and } 1 \leq i \leq n\} \\ & \cup \{o \text{ of arity } 0 \mid o \in O_0\} \cup \{\emptyset \text{ of arity } 0\}\end{aligned}$$

By $\mathcal{T}_\psi$ we will denote the set of ground terms.

For a command $c(x_1, \cdots, x_n) \in C$, we say that $x_i$, for some $1 \leq i \leq n$, is a *child argument* of $c$ if "create subject $x_i$" or "create object $x_i$" appears in $c$. Otherwise, $x_i$ is a *parent argument* of $c$. We define the relation $\equiv_Q$ over the objects of a configuration $Q = (S, O, P) \in Cf(R)$ as follows:

$o \equiv_Q o$ if $o \in O_0$;

$o \equiv_Q o'$ if $o$ and $o'$ were created as the $i$-th argument of a command $c$ applied with $o_{p_1}, \cdots, o_{p_m}$ as parent arguments for $o$ and with $o'_{p_1}, \cdots, o'_{p_m}$ as parent arguments for $o'$, and $o_{p_j} \equiv_Q o'_{p_j}$, for all $1 \leq j \leq m$. ($p_1, \cdots, p_m$ are the indexes of the parent arguments of $c$)

The fact that $o$ was created as the $i$-th argument of a command $c$ applied with $o_{p_1}, \cdots, o_{p_m}$ as parent arguments, can be memorized in a configuration $Q$ in many ways. For example, we can modify the system $\psi$ by adding a right *parent* and a right $c_i$, for all $c(x_1, \cdots, x_n) \in C$ and $1 \leq i \leq n$, and by transforming every command $c(x_1, \cdots, x_n) \in C$, such that after creating $x_i$, for some $1 \leq i \leq n$, we enter $c_i$ in $[x_i, x_i]$ and *parent* in $[x_{p_j}, x_i]$ for all $x_{p_j}$ parent arguments of $c$. In the following, for the simplicity of the exposition, we will not formalize this.

Clearly, the relation above is an equivalence and to every equivalence class we can uniquely associate a ground term from $\mathcal{T}_\psi$. Consequently, we

will denote an equivalence class by $[t]_Q$, where $t$ is the corresponding term from $\mathcal{T}_\psi$.

In the following, when we say equivalence relation we mean the relation $\equiv_Q$ and when we say equivalence class, we mean an equivalence class of $\equiv_Q$.

**Definition 2.19** Let $\psi = (R, C, Q_0)$ be a protection system. We say that a term from $\mathcal{T}_\psi$ is *accessible* if there exists $Q \in Cf(R)$ such that $Q_0 \rightarrow_\psi^* Q$ and $[t]_Q \neq \emptyset$. By $Acc(\psi)$ we will denote the set of accessible terms.

**Definition 2.20** A monotonic protection system $\psi = (R, C, Q_0)$ is called *creation-independent* if $R$ can be partitioned into two disjunctive sets $R_c$ and $R_e$ such that:

- the "create" commands test for and enter only rights from $R_c$;
- the other commands (the commands that contain only "enter" operations) enter only rights from $R_e$.

We can easily see that MTAM systems are particular cases of creation-independent protection systems in which $R_c = T$, $R_e = R$ and the tests for the rights in $R_c$ are as in command $\gamma(c)$ from Fig. 1.

If $\psi = (R, C, Q_0)$ is a protection system, we will denote by $Reach_\psi(Q, C')$, where $Q \in Cf(R)$ and $C' \subseteq C$, the set of reachable configurations from $Q$ using only commands from $C'$.

Now, we prove that for any creation-independent system $\psi$, any $t \in Acc(\psi)$ and any reachable configuration $Q$, we can apply in $Q$ a sequence of "create" commands to create an object from an equivalence class represented by $t$.

**Lemma 2.3** Let $\psi = (R, C, Q_0)$ be a creation-independent protection system and $C' \subseteq C$ the set of "create" commands. Then,

$$(\forall t \in Acc(\psi))(\forall Q \in Reach_\psi(Q_0, C)(\exists Q' \in Reach_\psi(Q, C'))(|[t]_{Q'}| = |[t]_Q|+1)$$

**Proof** From Definition 2.20, we can see that the application of a "create" command is not influenced in any way by the application of a command from $C - C'$.

Because $\psi$ is also monotonic we can easily obtain the result above. □

**Theorem 2.11** Let $\psi = (R, C, Q_0)$ be a creation-independent protection system. If $Acc(\psi)$ is finite then, $\psi$ belongs to $Dec$.

**Proof** If $\psi$ is a creation-independent protection system, then $R$ can be partitioned into $R_c$ and $R_e$ as in Definition 2.20.

Let $\psi_f = (R, C_f, Q'_0 = (S'_0, O'_0, P'_0))$ be a protection system, where $C_f \subseteq C$ is the set of commands that do not create objects and:

- $O'_0 = \{t | t \in Acc(\psi)\}$. Clearly, $O_0 \subseteq O'_0$;
- $S'_0$ is the set of subjects from $O'_0$;
- $P'_0$ is defined such as it's restriction to objects from $O_0$ is $P_0$ and in the cells of the other objects we have the rights from $R_c$ entered by the corresponding "create" commands.

In other words, $Q'_0$ is obtained from $Q_0$ applying "create" commands such that we obtain objects from equivalence classes represented by all terms in $Acc(\psi)$.

We will prove that $\psi$ is quasi-bisimilar to $\psi_f$ w.r.t $id_{O_0}$ and $id_R$.

In the following, for a configuration $Q = (S, O, P)$, $f_Q : O \to Acc(\psi)$ is a function such that $f(o) = t$ iff $o \in [t]_Q$.

We consider the following relation $B$: given $Q = (S, O, P)$ reachable in $\psi$ and $Q' = (S', O', P')$ configuration from $Cf(R)$, we have $B(Q, Q')$ if:

- $O_0 \subseteq O$ and $O' = Acc(\psi)$;
- $r \in P'(t_1, t_2)$ iff there exists $s \in [t_1]_Q$ and $o \in [t_2]_Q$ such that $r \in P(s, o)$.

To prove that $B$ is a quasi-bisimulation, let $Q$ and $Q'$ be two configurations like above such that $B(Q, Q')$.

Clearly, $Q \preceq_{id_{O_0}, id_R} Q'$.

Now, let $Q_1$ be a configuration such that $Q \to_\psi Q_1$. If we apply a "create" command then, we can find $Q'_1 = Q'$, such that $Q' \to^*_{\psi_f} Q'_1$ and $B(Q_1, Q'_1)$. Otherwise, suppose we apply a command $c(x_1, \cdots, x_n) \in C'$, with actual arguments $o_1, \cdots, o_n$. Since $B(Q, Q')$, we can apply $c(x_1, \cdots, x_n)$ with actual arguments $f_Q(o_1), \cdots, f_Q(o_n)$ in $Q'$ and obtain a configuration $Q'_1$ such that $B(Q_1, Q'_1)$.

For the reverse, suppose we have $Q' \to_{\psi_f} Q'_1$, for some configuration $Q'_1$. Clearly, in this step we apply a command $c(x_1, \cdots, x_n)$ that contain only "enter" primitive operations. Suppose it is applied using $t_1, \cdots, t_n \in Acc(\psi)$ as actual arguments.

Since $\psi$ is monotonic and create-independent, we can reach in $\psi$ from $Q$ a configuration $\overline{Q}$ such that $B(\overline{Q}, Q')$ and the access matrix of $\overline{Q}$ includes that of $Q$ and has in plus one object $o_i$ for each $t_i$ above, such that $o_i \in [t_i]_{\overline{Q}}$ and $o_i$ has in his cells the same rights as $t_i$ in $Q'$ (the creation of objects

is possible by Lemma 2.3 and the rights can be entered in the cells of $o_i$ because, once we have applied in $\psi$ a command that contains only "enter" primitive operations, we can apply it later eventually with equivalent actual arguments).

Now, in $\overline{Q}$ we can apply $c$ with $o_1,\cdots,o_n$ as actual arguments and obtain a configuration $Q_1$ such that $B(Q_1, Q_1')$.

The fact that $B(Q_0, Q_0')$ ends our proof. □

Using Theorem 2.11, we will prove that MTAM systems with acyclic creation graphs and mono-operational protection systems belong to *Dec*.

**Corollary 2.8** MTAM systems with acyclic creation graphs belong to *Dec*.

**Proof** As stated above we suppose that MTAM systems are in canonical form and consequently, they are creation-independent.

Since the creation graph is acyclic, we have also that the set of accessible terms is finite and we can apply Theorem 2.11, to obtain the statement of this corollary. □

**Mono-operational Protection Systems** Mono-operational protection systems ([63]) are protection systems with mono-operational commands. We will show that they are included in *Dec* in two steps, by proving first that the subclass of monotonic mono-operational protection systems is included in *Dec*.

**Theorem 2.12** Monotonic mono-operational protection systems belong to *Dec*.

**Proof** In the following we will consider protection systems such that every object is also a subject. This can be assumed without loss of generality by introducing an otherwise empty row for each pure object.

Hence, let $\psi = (R, C, Q_0 = (O_0, O_0, P_0)$ be a monotonic mono-operational protection system such that the commands in $C$ do not contain "create object" primitive operations.

We prove that $\psi$ is quasi-bisimilar to some monotonic monooperational system $\psi'$ that is creation-independent and has $Acc(\psi')$ finite. Consequently, by Theorem 2.11, $\psi' \in Dec$ and from the definition of *Dec*, $\psi \in Dec$.

Let $\psi' = (R \cup \{alive\}, C', Q_0' = (O_0, O_0, P_0'))$, where $P_0'$ is defined by:

$$P_0'(s,o) = \begin{cases} P_0(s,o) \cup \{alive\}, & \text{if } s = o; \\ P_0(s,o), & \text{otherwise.} \end{cases}$$

and $C'$ is obtained from $C$ in the following way:

- modify each conditional part of an "enter" command (command that contains an "enter" primitive operation) $c(x_1, \cdots, x_n) \in C$ by adding tests of the form: *alive* in $[x_i, x_i]$, for all $1 \leq i \leq n$;

- add a command $c_s(x)$ that has the conditional part empty and only a primitive operation "create subject $x$".

- remove each "create" command $c(x_1, \cdots, x_n)$ that creates a subject $x_i$, for some $1 \leq i \leq n$, and add an "enter" command with the conditional part of $c$, modified as in the first case, and a primitive operation "enter *alive* in $[x_i, x_i]$".

Now, we prove that $\psi \preceq_{id_{O_0}, id_R} \psi'$, using the following relation $B$: if $Q = (O, O, P) \in Cf(R)$ and $Q' = (O', O', P') \in Cf(R \cup \{alive\})$, we have $B(Q, Q')$ if:

- $O_0 \subseteq O$ and $O_0 \subseteq O'$;

- if $O'_{alive} = \{o | o \in O' \text{ and } alive \in P'(o, o)\}$ then, there exists a bijection $\phi : O \to O'_{alive}$, such that:

  - $\phi(o) = o$, for every $o \in O \cap O_0$;
  - $r \in P(o_1, o_2) \Leftrightarrow r \in P'(\phi(o_1), \phi(o_2))$, for all $o_1, o_2 \in S$ and $r \in R$.

We can easily prove that $B$ is a quasi-bisimulation relation if we take in consideration the following:

- applying a "create" command in $\psi$ is equivalent with applying in $\psi'$ the "create" command $c_s$ and an "enter" command that gives to this new subject the *alive* right to himself;

- in $\psi'$ we can create more subjects than in $\psi$, but if they do not have the *alive* right to themselves, they are useless. In fact, only to commands that enter the *alive* right in $\psi'$, we associate a "create" command in $\psi$.

Clearly, $\psi'$ is create-independent since the only "create" command does not test or enter any right. $Acc(\psi')$ is finite, because all the created objects in $\psi'$ are from an equivalence class represented by the same term $c_s(\emptyset)$. □

**Theorem 2.13** Mono-operational protection systems belong to *Dec*.

**Proof** Let $\psi = (R, C, Q_0)$ be a mono-operational system.

From Example 2.10 we have that $\psi \prec_{id_{O_0}, id_R} \psi_m$, where $\psi_m = (R, C_m, Q_0)$ is the monotonic restriction of $\psi$.

As $C_m \subseteq C$, from Example 2.9 we have also that $\psi_m \prec_{id_{O_0}, id_R} \psi$.

The fact that $id_{O_0}$ and $id_R$ are injective and $\psi_m$ is a monotonic mono-operational protection system, which by Theorem 2.12 belongs to *Dec*, concludes our proof. □

**Take-grant Systems** Take-grant systems ([78]) are protection systems $\psi = (R, C, Q_0 = (O_0, O_0, P_0))$, where $R = \{t, g, c\}$ and $C$ is the set of commands shown in Fig. 2.4, for all $\alpha, \beta, \gamma \in R$. It is clear that the system is monotonic and all objects are also subjects.

In the original paper, take-grant systems were presented as graph transformation systems. A configuration $Q = (O, O, P)$ is represented as a labeled directed graph, using subjects as nodes and cells in the matrix as labeled arcs (if $P(o_1, o_2) \neq \emptyset$, we have an arc from $o_1$ to $o_2$ labeled with $P(o_1, o_2)$ ). The commands in Fig. 2.4 are represented as graph transformations that introduce nodes and/or arcs.

We say that two nodes are *connected* if there exists a path between them, independent of the directionality or labels of the arcs. The decidability of the safety problem is obtained from the following theorem:

**Theorem**
*Let $\psi$ be a take-grant system. $\psi$ is leaky for $(o_1, o_2, r)$ iff in $\mathcal{G}_0$ (the graph that represents $Q_0$) $o_1$ and $o_2$ are connected and there exists an incoming arc in $o_2$ labeled with $t$ or $c$ if $r = t$ or with $r$ if $r \in \{g, c\}$.*

We will prove that take-grant protection systems are in *Dec*, by showing that they are quasi-bisimilar to a finite protection system with the same initial configuration.

**Theorem 2.14** Take-grant systems belong to *Dec*.

**Proof** If $\psi = (R, C, Q_0 = (O_0, O_0, P_0))$ is a take-grant protection system like above, let $\psi_f = (R, C', Q_0)$, where $C'$ contains all the commands of the following form:

command $c_{i,\alpha}(x, y, z, x_1, \cdots, x_i)$
  if connected$(x, y, x_1, \cdots, x_i)$
    $\alpha$ in $[z, y]$
  then
    enter $\alpha$ in $[x, y]$

where $0 \leq i \leq |O_0| - 2$ and $\alpha \in R$. connected$(x, y, x_1, \cdots, x_i)$ is a set of

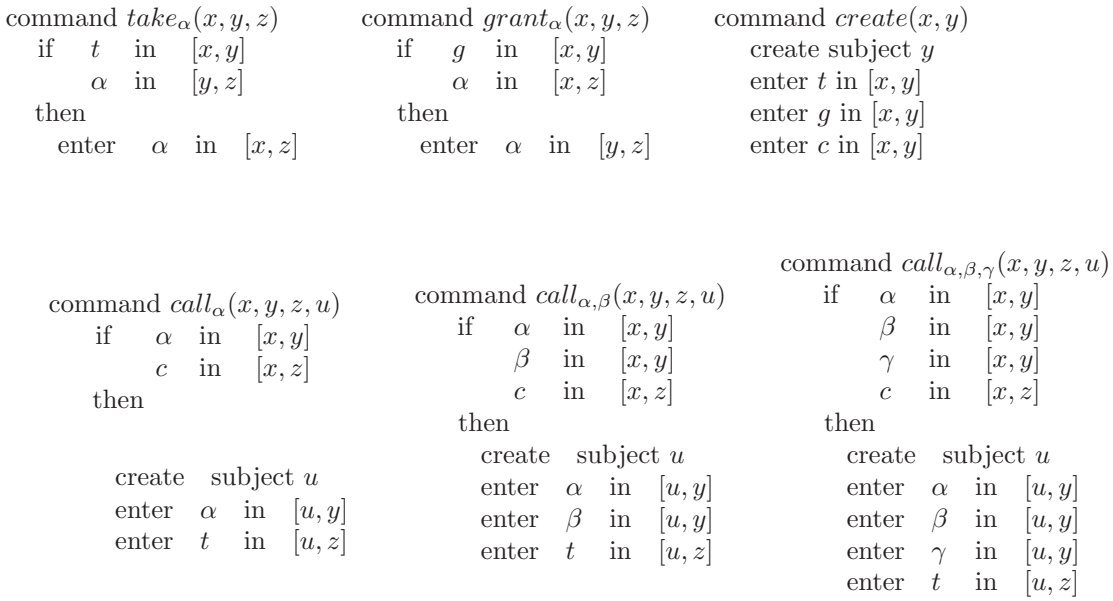


Figure 2.4: Take-grant commands.

conditions obtained from the conditions below, choosing one from each line:

$$\begin{array}{lll} \beta_1 \text{ in } [x, x_1] & \text{or} & \beta_1 \text{ in } [x_1, x] \\ \beta_2 \text{ in } [x_1, x_2] & \text{or} & \beta_2 \text{ in } [x_2, x_1] \\ & \cdots & \\ \beta_{i+1} \text{ in } [x_i, y] & \text{or} & \beta_{i+1} \text{ in } [y, x_i]. \end{array}$$

Above, $\beta_k$, for $1 \leq k \leq i+1$, can be any right from $R$.

Intuitively, connected$(x, y, x_1, \cdots, x_i)$ checks if in the graph that represents the configuration in which we apply $c_{i,\alpha}$, the nodes $x$ and $y$ are connected by a path of length $i+2$ that passes through $x_1$,$\cdots$,$x_i$.

We will prove that $\psi \preceq_{id_{O_0}, id_R} \psi_f$, considering the following relation $B$: given $Q_1 = (S_1, O_1, P_1)$ reachable from $Q_0$ and $Q_2 = (S_2, O_2, P_2) \in Cf(R)$, we have $B(Q_1, Q_2)$ if:

- $O_2 = O_0$;
- $r \in P_1(s, o) \Leftrightarrow r \in P_2(s, o)$, for any $s \in S_0$, $o \in O_0$ and $r \in R$.

Now, we will prove that $B$ is a quasi-bisimulation relation between $\psi$ and $\psi_f$ w.r.t. $id_{O_0}$ and $id_R$.

$Q_1 \preceq_{id_{O_0}, id_R} Q_2$ is straightforward from the definition of $B$.

Now suppose that $Q_1 \rightarrow_\psi Q_1'$ by a command $c$.

If $c$ is $take_\alpha$, then suppose it is applied with some actual arguments $s_1$, $s_2$ and $s_3$. If all these objects are initial then, because $c$ is also present in $C'$, we can apply it with the same actual arguments in $Q_2$ and obtain a configuration $Q_2'$ such that $B(Q_1', Q_2')$.

If not, we have two cases: whether or not $s_1$ and $s_3$ are both initial objects. If they are not both initial objects then, we can find $Q_2' = Q_2$ such that $Q_2 \rightarrow_{\psi_f}^* Q_2'$ and $B(Q_1', Q_2')$.

If $s_1$ and $s_3$ are both from $O_0$ then from the main result of [78] stated above, we have that there exists some initial objects $o_1$,$\cdots$,$o_i$, for some $i$ between 0 and $|O_0| - 2$, such that $connected(s_1, s_3, o_1, \cdots, o_i)$ is true and also, there exists some initial object $o$ such that $\alpha \in P_0(o, s_3)$. Since $\psi$ and $\psi_f$ are monotonic, these conditions are true also in $Q_2$ and thus, we can apply a command from $C'$ to add $\alpha$ in $[s_1, s_3]$. Consequently, we can obtain a configuration $Q_2'$ such that $Q_2 \rightarrow_{\psi_f}^* Q_2'$ and $B(Q_1', Q_2')$.

The case when $c$ is $grant_\alpha$ is similar.

If $c$ is a *create* or *call* command then, we can find $Q_2' = Q_2$ such that $Q_2 \rightarrow_{\psi_f}^* Q_2'$ and $B(Q_1', Q_2')$.

For the reverse, suppose that $Q_2 \rightarrow_{\psi_f} Q_2'$. Also, from the main result of [78], we can find a configuration $Q_1'$ such that $Q_1 \rightarrow_{\psi}^* Q_1'$ and $B(Q_1', Q_2')$.

The fact that $B(Q_0, Q_0)$ concludes our proof. □

### 2.3.2 Abstractions and Preservation Results

We present abstractions of multi-valued Kripke structures and preservation results for $CTL^*$ formulas as introduced in [47]. The abstract system is obtained by applying equivalence relations on the state space of the concrete system. Then, the predicate symbols of the logic are re-defined to work properly on equivalence classes using interpretation policies as we have explained in Section 2.1.

**Definition 2.21** Let $M = (Q, R, L)$ be an mv-Kripke structure over $AP$ and $\mathcal{B}$, $\rho$ an equivalence relation on $Q$, and $\alpha_R$ and $\alpha_L$ two interpretation policies over $\mathcal{B}$. An mv-Kripke structure $M' = (Q', R', L')$ over $AP$ and $\mathcal{B}$ is called an $(\alpha_R, \alpha_L)$-*abstraction of* $M$ *by* $\rho$ if:

- $Q' = Q/_\rho$;
- $R'$ is the reinterpretation of $R$ on $Q' \times Q'$ according to $\alpha_R$;
- $L'$ is the reinterpretation of $L$ over $Q'$ according to $\alpha_L$.

The abstractions for Kripke structures introduced in [46] are instances of the Definition 2.21.

Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, and $D \subseteq B$. We say that a path $\pi_2 \in \mathsf{Paths}(M_2, D)$ is a *corresponding path* to $\pi_1 \in \mathsf{Paths}(M_1, D)$ if:

- $|\pi_2| = |\pi_1|$;
- $\pi_2(i) = [\pi_1(i)]$, for any $0 \leq i < |\pi_2|$.

We denote by $C_{M_1}(\pi_2)$ the set of all $D$-paths in $M_1$ that have $\pi_2$ as a corresponding $D$-path in $M_2$.

Let $\mathcal{B} = (B, \wedge, \vee, \neg)$ be a truth algebra and $D \subseteq B$. The set $D$ is called

- (*upward*) *closed under* $\leq$ if $b' \in D$ whenever $b \leq b'$ for some $b \in D$;
- *closed under lub* (*glb*) if $D$ contains the lub (glb) of any non-empty subset of elements in $D$;
- *backward closed under lub* (*glb*) if it includes any non-empty subset $X \subseteq B$ whenever it contains $lub(X)$ ($glb(X)$).

It is easy to see that $D$ is closed under lub and backward closed under glb, whenever it is closed under $\leq$.

Now we are in a position to establish several preservation results. The first one shows preservation results for subsets $D$ of truth values.

**Theorem 2.15** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, let $D \subseteq B$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL^*_+$ formula over $AP$. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \in D$ implies $L_1(q)(p) \in D$;
3. $D$ is closed under $\leq$ and glb, and backward closed under lub,

then

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \in D \quad \Rightarrow \quad [\phi]^{M_1}_{q} \in D)$$

and

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]^{M_2}_{\pi_2} \in D \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\psi]^{M_1}_{\pi_1} \in D)).$$

**Proof** We will prove the statements in the theorem by simultaneous structural induction on the formulas $\phi$ and $\psi$. The following cases are to be considered:

- $\phi = p \in AP$. Let $q \in Q_1$. If we assume that $[\phi]^{M_2}_{[q]} \in D$, then $L_2([q])(p) \in D$ and by 2 we obtain $L_1(q)(p) \in D$ which shows that $[\phi]^{M_1}_q \in D$;

- $\phi = \phi_1 \wedge \phi_2$, where $\phi_1$ and $\phi_2$ are state formulas. Given $q \in Q_1$ assume that $[\phi]^{M_2}_{[q]} \in D$ and both $\phi_1$ and $\phi_2$ satisfy the property in the theorem. Then, $[\phi_1]^{M_2}_{[q]} \wedge [\phi_2]^{M_2}_{[q]} \in D$. As $D$ is closed under $\leq$, $[\phi_1]^{M_2}_{[q]} \wedge [\phi_2]^{M_2}_{[q]} \leq [\phi_1]^{M_2}_{[q]}$ and $[\phi_1]^{M_2}_{[q]} \wedge [\phi_2]^{M_2}_{[q]} \leq [\phi_2]^{M_2}_{[q]}$, both $[\phi_1]^{M_2}_{[q]}$ and $[\phi_2]^{M_2}_{[q]}$ are in $D$ and by the induction hypothesis we obtain $[\phi_1]^{M_1}_q, [\phi_2]^{M_1}_q \in D$. Now we use the fact that $D$ is closed under glb and we get $[\phi_1]^{M_1}_q \wedge [\phi_2]^{M_1}_q \in D$ which shows that $[\phi]^{M_1}_q \in D$;

- the case $\phi = \phi_1 \vee \phi_2$, where $\phi_1$ and $\phi_2$ are state formulas, is similar to the previous one. One has to use backward closeness of $D$ under lub and then closeness of $D$ under $\leq$;

- $\phi = \forall\psi$, where $\psi$ is a path formula. Given $q \in Q_1$ assume that $[\phi]^{M_2}_{[q]} \in D$ and $\psi$ satisfies the property in the lemma. Then,

$$\bigwedge_{\pi_2 \in \mathsf{Paths}(M_2, D, [q])} [\psi]^{M_2}_{\pi_2} \in D$$

  and because $D$ is closed under $\leq$ we obtain $[\psi]^{M_2}_{\pi_2} \in D$, for any $\pi_2 \in \mathsf{Paths}(M_2, D, [q])$.

  Let $\pi_1$ be a $D$-path from $q$ in $M_1$. According to 1, there exists a corresponding $D$-path $\pi_2$ from $[q]$ in $M_2$. As $[\psi]^{M_2}_{\pi_2} \in D$, the induction hypothesis leads to $[\psi]^{M_2}_{\pi_1} \in D$ and the closeness of $D$ under glb concludes the case by showing that

$$[\phi]^{M_1}_q = \bigwedge_{\pi_1 \in \mathsf{Paths}(M_1, D, q)} [\psi]^{M_1}_{\pi_1} \in D;$$

- $\psi = \psi_1 \vee \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas. Given $\pi_2$ a $D$-path in $M_2$ starting at some state $[q]$ assume that $[\psi]^{M_2}_{\pi_2} \in D$ and $\psi_1$ and $\psi_2$ satisfy the property in the theorem. Then, by the fact that $D$ is backward closed under lub, both $[\psi_1]^{M_2}_{\pi_2}$ and $[\psi_2]^{M_2}_{\pi_2}$ are in $D$. By the induction hypothesis, $[\psi_1]^{M_1}_{\pi_1}, [\psi_2]^{M_1}_{\pi_1} \in D$, for any $\pi_1 \in C_{M_1}(\pi_2)$. Using the closeness of $D$ under $\leq$, we obtain $[\psi_1]^{M_1}_{\pi_1} \vee [\psi_2]^{M_1}_{\pi_1} \in D$, that is $[\psi]^{M_1}_{\pi_1} \in D$, for any $\pi_1 \in C_{M_1}(\pi_2)$;

- the case $\psi = \psi_1 \wedge \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas, is similar to the previous case;

- $\psi = \mathbf{X}\psi_1$, where $\psi_1$ is a path formula. Given $\pi_2$ a $D$-path in $M_2$ starting at some state $[q]$ assume that $[\psi]^{M_2}_{\pi_2} \in D$ and $\psi_1$ satisfies the property in the theorem. Then, $|\pi_2| > 1$ and

  $$R_2(\pi_2(0), \pi_2(1)) \wedge [\psi_1]^{M_2}_{\pi_2^1} \in D,$$

  which by the backward closeness of $D$ under glb implies $[\psi_1]^{M_2}_{\pi_2^1} \in D$.

  Let $\pi_1 \in C_{M_1}(\pi_2)$. Clearly, $\pi_1^1 \in C_{M_1}(\pi_2^1)$ and, therefore, $[\psi_1]^{M_1}_{\pi_1^1} \in D$ by the induction hypothesis. As a conclusion, $[\psi]^{M_1}_{\pi_1} \in D$;

- the case $\psi = \overline{\mathbf{X}}\psi_1$, where $\psi_1$ is a path formula is similar to the previous one;

- the cases $\psi = \psi_1 \,\mathbf{U}\, \psi_2$ and $\psi = \psi_1 \,\mathbf{R}\, \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas, purports a similar discussion as those above. □

**Remark 2.3** The constraints the set $D$ in Theorem 2.15 should satisfy are similar to the ones in Theorem 1 in [73]. The main difference is that in [73] there is a bisimulation between an mv-Kripke structure and a standard Kripke structure, while Theorem 2.15 in this paper is based on a simulation from an mv-Kripke structure to another mv-Kripke structure.

**Remark 2.4** The first two conditions in Theorem 2.15 can be proved looking only at the interpretation policies used in the abstraction. Hence, the first condition holds if

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

for any $b \in B - D$, while the second holds if

$$(\alpha_L(d) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_L(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq D),$$

for any $d \in D$.

Preserving punctual truth values $b \in B$ is harder than preserving subsets of truth values. We present below a suite of results along this line. As we will see, we have to distinguish between $\forall CTL^*$ formulas and $\exists CTL^*$ formulas.

**Theorem 2.16** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, $D \subseteq B$, $b \in B$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL^*_+$ formula over $AP$. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \geq b$ implies $L_1(q)(p) \geq b$;
3. for any $q, q' \in Q_1$, $R_2([q], [q']) \geq b$ implies $R_1(q, q') \geq b$;
4. for any subset $B'$ of $B$, $\vee B' \geq b$ implies $b' \geq b$ for some $b' \in B'$,

then:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \geq b \quad \Rightarrow \quad [\phi]^{M_1}_{q} \geq b)$$

and

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]^{M_2}_{\pi_2} \geq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \geq b)).$$

**Proof** We will prove the statements in the theorem by simultaneous structural induction on the formulas $\phi$ and $\psi$. The following cases are to be considered:

- $\phi = p \in AP$. Let $q \in Q_1$. If we assume that $[\phi]^{M_2}_{[q]} \geq b$, then $L_2([q])(p) \geq b$ and by 2 we obtain $L_1(q)(p) \geq b$ which shows that $[\phi]^{M_1}_{q} \geq b$;
- $\phi = \phi_1 \wedge \phi_2$, where $\phi_1$ and $\phi_2$ are state formulas. Given $q \in Q_1$ assume that $[\phi]^{M_2}_{[q]} \geq b$ and both $\phi_1$ and $\phi_2$ satisfy the property in the lemma. Then, $[\phi_1]^{M_2}_{[q]} \wedge [\phi_2]^{M_2}_{[q]} \geq b$ which leads to $[\phi_1]^{M_2}_{[q]} \geq b$ and $[\phi_2]^{M_2}_{[q]} \geq b$. By the induction hypothesis we obtain $[\phi_1]^{M_1}_{q}, [\phi_2]^{M_1}_{q} \geq b$. Consequently, $[\phi_1]^{M_1}_{q} \wedge [\phi_2]^{M_1}_{q} \geq b$ which shows that $[\phi]^{M_1}_{q} \geq b$;
- $\phi = \phi_1 \vee \phi_2$, where $\phi_1$ and $\phi_2$ are state formulas. Given $q \in Q_1$ assume that $[\phi]^{M_2}_{[q]} \geq b$ and both $\phi_1$ and $\phi_2$ satisfy the property in the lemma. Then, $[\phi_1]^{M_2}_{[q]} \vee [\phi_2]^{M_2}_{[q]} \geq b$ and by 4 we may assume that $[\phi_1]^{M_2}_{[q]} \geq b$. By the induction hypothesis we obtain $[\phi_1]^{M_1}_{q} \geq b$ which implies $[\phi]^{M_1}_{q} = [\phi_1]^{M_1}_{q} \vee [\phi_2]^{M_1}_{q} \geq b$;
- $\phi = \forall \psi$, where $\psi$ is a path formula. Given $q \in Q_1$ assume that $[\phi]^{M_2}_{[q]} \geq b$ and $\psi$ satisfies the property in the lemma. Then,

$$\bigwedge_{\pi_2 \in \mathsf{Paths}(M_2, D, [q])} [\psi]^{M_2}_{\pi_2} \geq b$$

and we obtain $[\psi]^{M_2}_{\pi_2} \geq b$, for all $\pi_2 \in \mathsf{Paths}(M_2, D, [q])$.

Let $\pi_1$ be a $D$-path from $q$ in $M_1$. According to 1, there exists a corresponding $D$-path $\pi_2$ from $[q]$ in $M_2$. As $[\psi]^{M_2}_{\pi_2} \geq b$, the induction hypothesis leads to $[\psi]^{M_2}_{\pi_1} \geq b$ and thus

$$[\phi]^{M_1}_q = \bigwedge_{\pi_1 \in \mathsf{Paths}(M_1, D, q)} [\psi]^{M_1}_{\pi_1} \geq b;$$

- the cases $\psi = \psi_1 \wedge \psi_2$ and $\psi = \psi_1 \vee \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas, can be discussed as the similar cases for state formulas;
- $\psi = \mathbf{X}\psi_1$, where $\psi_1$ is a path formula. Given $\pi_2$ a $D$-path in $M_2$ starting at some state $[q]$ assume that $[\psi]^{M_2}_{\pi_2} \geq b$ and $\psi_1$ satisfies the property in the lemma. Then, $|\pi_2| > 1$ and

  $$R_2(\pi_2(0), \pi_2(1)) \wedge [\psi_1]^{M_2}_{\pi_2^1} \geq b$$

  which leads to $R_2(\pi_2(0), \pi_2(1)) \geq b$ and $[\psi_1]^{M_2}_{\pi_2^1} \geq b$.

  Let $\pi_1 \in C_{M_1}(\pi_2)$. Clearly, $\pi_1^1 \in C_{M_1}(\pi_2^1)$ and, therefore, $[\psi_1]^{M_1}_{\pi_1^1} \geq b$ by the induction hypothesis. Combining this with $R_1(\pi_1(0), \pi_1(1)) \geq b$ which follows from 3, we obtain $[\psi]^{M_1}_{\pi_1} \geq b$;
- the cases $\psi = \overline{\mathbf{X}}\psi_1$, $\psi = \psi_1 \,\mathbf{U}\, \psi_2$, and $\psi = \psi_1 \,\mathbf{R}\, \psi_2$, where $\psi_1$ and $\psi_2$ are path formulas, purport a similar discussion as those above. □

**Remark 2.5** As it was the case with the previous theorem, the first three conditions in Theorem 2.16 can be proved looking only at the interpretation policies used in the abstraction. Hence, the first condition holds if

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

for any $b \in B - D$, while the second and the third hold if

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow),$$

for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \geq b$.

The preservation results for Kripke structures over Kleene's three-valued interpretation ($\mathcal{B}_3$ denotes the corresponding truth algebra) from Lemmas 1 and 2, Section 5, in [46] can be obtained as a particular case of Theorem 2.16.

**Corollary 2.9** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B}_3$, $D = \{\bot, 1\}$, and $\phi$ be a mv-$LTL_+$ formula over $AP$. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \geq \bot$ implies $L_1(q)(p) \geq \bot$,

then

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \geq \bot \quad \Rightarrow \quad [\phi]^{M_1}_q \geq \bot).$$

**Proof** Directly from Theorem 2.16 (the third condition can be discarded because $D = \{\bot, 1\}$). □

**Corollary 2.10** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B}_3$, $D = \{\bot, 1\}$, and $\phi$ be a mv-$LTL_+$ formula over $AP$. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q, q' \in Q_1$, $R_2([q], [q']) = 1$ implies $R_1(q, q') = 1$;
3. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 1$ implies $L_1(q)(p) = 1$,

then

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} = 1 \quad \Rightarrow \quad [\phi]^{M_1}_q = 1).$$

Moreover, the result holds if the second condition is replaced by $R_1(q, q') \in \{0, 1\}$, for all $q, q' \in Q_1$.

**Proof** The first part follows directly from Theorem 2.16 and the second part can be proved in a similar manner. □

The following preservation result for abstractions of Kripke structures over Kleene's three-valued interpretation has been introduced in [46] but it depends too much on the particular structure of the truth algebra $\mathcal{B}_3$ to be deduced from some general preservation result.

**Theorem 2.17** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B}_3$, $D = \{\bot, 1\}$, and $\phi$ be a mv-$LTL_+$ formula over $AP$. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, D)$ there exists a $D$-path $\pi_1 \in C_{M_1}(\pi_2)$;

2. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;

3. for any $q, q' \in Q_1$, $R_2([q], [q']) = \bot$ implies $R_1(q, q') = \bot$;

4. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = \bot$ implies $L_1(q)(p) = \bot$,

5. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 1$ implies $L_1(q)(p) \in \{\bot, 1\}$,

then

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} = \bot \quad \Rightarrow \quad [\phi]^{M_1}_{q} = \bot).$$

**Proof** By Corollary 2.9 we obtain that

$$[\phi]^{M_2}_{[q]} = \bot \Rightarrow [\phi]^{M_1}_{q} \in \{1, \bot\},$$

for any $\phi \in LTL_+$ and $q \in Q_1$. What remains to be proved is that $[\phi]^{M_1}_{q} = \bot$.

According to the semantics of $LTL_+$, the fact that

$$[\phi]^{M_2}_{[q]} = \bigwedge_{\pi \in Path(M_2, D, [q])} [\phi]^{M_2}_{\pi} = \bot$$

implies that there exists $\pi_2$ a path in $M_2$ starting at $[q]$ such that $[\phi]^{M_2}_{\pi_2} = \bot$.

By 2, there exists a path $\pi_1$ in $M_1$ starting at $q$ such that $\pi_1(i) \in \pi_2(i)$, for any $0 \leq i < |\pi_1|$. We will prove that $[\phi]^{M_2}_{\pi_2} = \bot$ implies $[\phi]^{M_1}_{\pi_1} = \bot$, which concludes the proof.

Similarly to the proof of Corollary 2.9, by structural induction on $\phi$ we can easily show that $[\phi]^{M_2}_{\pi_2} \in \{1, \bot\}$ implies $[\phi]^{M_1}_{\pi_1} \in \{1, \bot\}$.

Now, by structural induction on $\phi$ we show that $[\phi]^{M_2}_{\pi_2} = \bot$ implies $[\phi]^{M_1}_{\pi_1} = \bot$:

- $\phi = p \in AP$. This case follows directly from 4;

- $\phi = \psi_1 \wedge \psi_2$. Assume that $\psi_1$ and $\psi_2$ satisfy the property. Then, $[\psi_1 \wedge \psi_2]^{M_2}_{\pi_2} = [\psi_1]^{M_2}_{\pi_2} \wedge [\psi_2]^{M_2}_{\pi_2} = \bot$ implies $[\psi_1]^{M_2}_{\pi_2}, [\psi_2]^{M_2}_{\pi_2} \in \{1, \bot\}$, and $[\psi_1]^{M_2}_{\pi_2} = \bot$ or $[\psi_2]^{M_2}_{\pi_2} = \bot$. Then we get $[\psi_1]^{M_1}_{\pi_1}, [\psi_2]^{M_1}_{\pi_1} \in \{1, \bot\}$, and from the induction hypothesis it follwos $[\psi_1]^{M_1}_{\pi_1} = \bot$ or $[\psi_2]^{M_1}_{\pi_1} = \bot$. Hence, $[\phi]^{M_1}_{\pi_1} = \bot$;

- $\phi = \psi_1 \vee \psi_2$. This is similar to the previous case;

- $\phi = \mathbf{X}\psi$. Assume that $\psi$ satisfies the property. Then, $[\mathbf{X}\psi]_{\pi_2}^{M_2} = |\pi_2| > 1 \wedge [\psi]_{\pi_2^1}^{M_2} \wedge R_2(\pi_2(0), \pi_2(1)) = \bot$ implies $|\pi_2| > 1$ and $[\psi]_{\pi_2^1}^{M_2} \wedge R_2(\pi_2(0), \pi_2(1)) = \bot$. But then, $[\psi]_{\pi_1^1}^{M_1}, R_1(\pi_1(0), \pi_1(1)) \in \{1, \bot\}$. If $[\psi]_{\pi_2^1}^{M_2} = \bot$, then, by the induction hypothesis, $[\psi]_{\pi_1^1}^{M_1} = \bot$ and we are done with this case. Otherwise, we use 3 and we are also done;
- the cases $\phi = \overline{\mathbf{X}}\psi$, $\phi = \psi_1 \mathbf{U} \psi_2$, and $\phi = \psi_1 \mathbf{R} \psi_2$ are discussed similarly and, therefore, they are omitted. □

**Theorem 2.18** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, $D \subseteq B$, $b \in B$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL_+^*$ formula over $AP$. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, D)$ there exists a $D$-path $\pi_1 \in C_{M_1}(\pi_2)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \leq b$ implies $L_1(q)(p) \leq b$;
3. for any $q, q' \in Q_1$, $R_2([q], [q']) \leq b$ implies $R_1(q, q') \leq b$;
4. for any subset $B'$ of $B$, $\wedge B' \leq b$ implies $b' \leq b$ for some $b' \in B'$,

then:

$$(\forall q \in Q_1)([\phi]_{[q]}^{M_2} \leq b \quad \Rightarrow \quad [\phi]_q^{M_1} \leq b)$$

and

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]_{\pi_2}^{M_2} \leq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]_{\pi_1}^{M_1} \leq b)).$$

**Proof** Almost all cases are pretty much the same as for Theorem 2.16 and, therefore, they are omitted. We only emphasize the following two cases:

- $\phi = \forall\psi$, where $\psi$ is a path formula. Given $q \in Q_1$ assume that $[\phi]_{[q]}^{M_2} \leq b$ and $\psi$ satisfies the property in the lemma. Then,

  $$\bigwedge_{\pi_2 \in \mathsf{Paths}(M_2, D, [q])} [\psi]_{\pi_2}^{M_2} \leq b$$

  and, by 4, we obtain $[\psi]_{\pi_2}^{M_2} \leq b$, for some $\pi_2 \in \mathsf{Paths}(M_2, D, [q])$.

  According to 1, there exists a corresponding $D$-path $\pi_1$ from $q$ in $M_1$. As $[\psi]_{\pi_2}^{M_2} \leq b$, the induction hypothesis leads to $[\psi]_{\pi_1}^{M_2} \leq b$ and thus

  $$[\phi]_q^{M_1} = \bigwedge_{\pi_1 \in \mathsf{Paths}(M_1, D, q)} [\psi]_{\pi_1}^{M_1} \leq b;$$

- the case "$\psi = \mathbf{X}\psi_1$" has to be split into two sub-cases: one corresponding to $|\pi_2| > 1$, which is handled as in the proof of Theorem 2.16, and one corresponding to $|\pi_2| \leq 1$. The later sub-case leads easily to $[\mathbf{X}\psi_1]_{\pi_2}^{M_2} = 0$ which implies $[\mathbf{X}\psi_1]_{\pi_1}^{M_1} = 0$ (see the notation in the proof of Theorem 2.16). □

**Remark 2.6** Again, the first condition in Theorem 2.18 holds if

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_R(b) = \exists_a^S \;\Rightarrow\; S \cup b\downarrow \cup b\uparrow \subseteq D),$$

for any $b \in D$, while the second and the third hold if

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \;\wedge\; (\alpha(d) = \exists_a^S \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow),$$

for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \leq b$.

We can obtain as a direct corollary of Theorem 2.18 the preservation result for Kripke structures over Kleene's three-valued interpretation from Lemma 3, Section 5, in [46].

**Corollary 2.11** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B}_3$, $D = \{\bot, 1\}$, and $\phi$ be a mv-$LTL_+$ formula over $AP$. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, D)$ there exists a $D$-path $\pi_1 \in C_{M_1}(\pi_2)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 0$ implies $L_1(q)(p) = 0$,

then

$$(\forall q \in Q_1)([\phi]_{[q]}^{M_2} = 0 \;\;\Rightarrow\;\; [\phi]_q^{M_1} = 0).$$

**Proof** Directly from Theorem 2.18 (the third condition can be discarded because $D = \{\bot, 1\}$). □

A similar preservation result have been obtained in [46] for $LTL_+^\infty$, the subset of $LTL$ consisting of formulas without negation and $\overline{\mathbf{X}}$.

**Corollary 2.12** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B}_3$, $D = \{\bot, 1\}$, and $\phi$ be a mv-$LTL_+^\infty$ formula over $AP$. If

1. for any $q \in Q_1$ and any path $\pi_2$ in $M_2$ starting at $[q]$, there exists a path $\pi_1$ in $M_1$ starting at $q$ with $|\pi_1| \leq |\pi_2|$ and $\pi_1(i) \in \pi_2(i)$, for any $0 \leq i < |\pi_1|$;

2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 0$ implies $L_1(q)(p) = 0$,

then

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} = 0 \quad \Rightarrow \quad [\phi]^{M_1}_{q} = 0).$$

**Theorem 2.19** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, $D \subseteq B$, $b \in B$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\exists CTL^*_+$ formula over $AP$. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, D)$ there exists a $D$-path $\pi_1 \in C_{M_1}(\pi_2)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \geq b$ implies $L_1(q)(p) \geq b$;
3. for any $q, q' \in Q_1$, $R_2([q], [q']) \geq b$ implies $R_1(q, q') \geq b$;
4. for any subset $B'$ of $B$, $\vee B' \geq b$ implies $b' \geq b$ for some $b' \in B'$,

then:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \geq b \quad \Rightarrow \quad [\phi]^{M_1}_{q} \geq b)$$

and

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]^{M_2}_{\pi_2} \geq b \quad \Rightarrow \quad (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \geq b)).$$

**Proof** Similar to the proof of Theorem 2.16. □

**Remark 2.7** Again, the first condition in Theorem 2.19 holds if

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cup b\downarrow \cup b\uparrow \subseteq D),$$

for any $b \in D$ while the second and the third hold if

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow),$$

for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \geq b$.

**Theorem 2.20** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, $D \subseteq B$, $b \in B$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\exists CTL^*_+$ formula over $AP$. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \leq b$ implies $L_1(q)(p) \leq b$;

3. for any $q, q' \in Q_1$, $R_2([q],[q']) \leq b$ implies $R_1(q,q') \leq b$;

4. for any subset $B'$ of $B$, $\wedge B' \leq b$ implies $b' \leq b$ for some $b' \in B'$,

then:

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} \leq b \;\; \Rightarrow \;\; [\phi]^{M_1}_{q} \leq b)$$

and

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]^{M_2}_{\pi_2} \leq b \;\; \Rightarrow \;\; (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{\pi_1} \leq b)).$$

**Proof** Similar to the proof of Theorem 2.18. □

**Remark 2.8** Again, the first condition in Theorem 2.20 holds if

$$(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \wedge (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset),$$

for any $b \in B - D$ while the second and the third hold if

$$(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow),$$

for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \leq b$.

**Remark 2.9** Let $M_2 = (Q_2, R_2, L_2)$ be an $(\alpha_R, \alpha_L)$-abstraction by an equivalence $\rho$ of an mv-Kripke structure $M_1 = (Q_1, R_1, L_1)$ over $AP$ and $\mathcal{B} = (B, \leq)$, $D \subseteq B$ and $b \in B$. If all the conditions from Theorems 2.16, 2.18, 2.19 and 2.20 hold and also,

- for any $q' \in Q_1$ and any $p \in AP$, $L_2([q'])(p) = b$ implies $L_1(q')(p) = b$;
- for any $q', q'' \in Q_1$, $R_2([q'],[q'']) = b$ implies $R_1(q',q'') = b$;
- for any subset $B'$ of $B$, $\wedge B' = b$ implies $b \in B'$;
- for any subset $B'$ of $B$, $\vee B' = b$ implies $b \in B'$;

then,

$$(\forall q \in Q_1)([\phi]^{M_2}_{[q]} = b \;\Rightarrow\; [\phi]^{M_1}_{q} = b) \text{ and}$$

$$(\forall \pi_2 \in \mathsf{Paths}(M_2, D))([\psi]^{M_2}_{\pi_2} = b \;\Rightarrow\; (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{M_1}_{q} = b)),$$

for any $\phi$ and $\psi$ a state and, respectively, a path mv-$CTL^*_+$ formula over $AP$.

### 2.3.3 Relating abstractions

When we want to verify a system by abstraction, first we choose an equivalence and then a type of abstraction (a pair of interpretation policies $(\alpha_R, \alpha_L)$). The abstraction might not allow us to draw any conclusion (about the property we want to check) and, therefore, we might try to change the abstraction: either the equivalence, or the type of abstraction, or both of them. The simplest choice is to change the type of abstraction. In such a case it would be very helpful to know the relationships between types of abstraction in order to avoid those types which would lead us to the same conclusions as the previous type of abstraction.

In this section we will study the relationships between the types of abstraction of Kripke structures over Kleene's three-valued interpretations obtained using the following interpretation policies [46]:

- $\alpha_1(0) = \forall$, $\alpha_1(\bot) = \exists_a^{\{0,\bot,1\}}$ and $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \exists^{\{0,\bot,1\}}$, $\alpha_2(\bot) = \exists^{\{\bot,1\}}$ and $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\bot) = \exists^{\{0,\bot\}}$ and $\alpha_3(1) = \exists^{\{0,\bot,1\}}$.

**Theorem 2.21** Let $M_1$ be an mv-Kripke structure over some set of atomic propositions $AP$ and $\mathcal{B}_3$, $M_2$ an $(\alpha_2, \alpha_3)$- or a $(\alpha_2, \alpha_1)$-abstraction of $M_1$ based on some equivalence $\rho$, and $M_3$ an $(\alpha_3, \alpha_2)$-abstraction of $M_1$ based on $\rho$. Then,

$$[\phi]_{[q]}^{M_2} = 0 \Rightarrow [\phi]_{[q]}^{M_3} = 0,$$

for any $\phi \in LTL_+$ and $q \in Q_1$.

**Proof** Assume that $M_2$ is a $(\alpha_2, \alpha_3)$-abstraction of $M_1$ (the other case can be obtained analogously). Clearly, $Q_2 = Q_3/\rho'$, where $\rho'$ is the identity.

Notice that Corollary 2.11 takes place even if $Q_2 = Q_1/\rho$ and $M_2$ is not necessarily an $(\alpha_R, \alpha_L)$-abstraction of $M_1$.

We prove that the hypothesis of Corollary 2.11 hold for $M_3$ and $M_2$ (in this order). Let $\pi_2$ be a path in $M_2$ starting at $[q]_\rho$, for some $q \in Q_1$. By the definition of $R_2$, there exists a path $\pi_1$ in $M_1$ starting at $q$ such that $\pi_2(i) = [\pi_1(i)]_\rho$, for any $0 \leq i < |\pi_2|$. Now, by the definition of $R_3$, there exists a path $\pi_3$ in $M_3$ starting at $[q]_\rho$ such that $\pi_1(i) \in \pi_3(i)$, for any $0 \leq i < |\pi_1|$. Since $\pi_1(i)$ can be in a unique abstract state with respect to $\rho$, we obtain that $\pi_3(i) = \pi_2(i)$, for all $0 \leq i < |\pi_2|$. From the construction above we also obtain $|\pi_2| = |\pi_3|$. Now, if $L_2([q]_\rho, p) = 0$, for some $q \in Q_1$, then $L_1(q_1, p) = 0$, for any $q_1 \in [q]_\rho$ and, consequently, $L_3([q]_\rho, p) = 0$ which concludes the proof. □

**Theorem 2.22** Let $M_1$ be an mv-Kripke structure over some set of atomic propositions $AP$ and $\mathcal{B}_3$, $M_2$ an $(\alpha_3, \alpha_2)$-abstraction of $M_1$ based on some equivalence $\rho$, and $M_3$ an $(\alpha_1, \alpha_2)$- or an $(\alpha_1, \alpha_1)$-abstraction of $M_1$ based on $\rho$. Then,

1. $[\phi]^{M_2}_{[q]} = \bot \Rightarrow [\phi]^{M_3}_{[q]} = \bot$,
2. $[\phi]^{M_3}_{[q]} = 1 \Rightarrow [\phi]^{M_2}_{[q]} = 1$,

for any $\phi \in LTL_+$ and $q \in Q_1$. Moreover, if $M_3$ is an $(\alpha_1, \alpha_2)$-abstraction then,

3. $[\phi]^{M_3}_{[q]} = \bot \Rightarrow [\phi]^{M_2}_{[q]} \in \{\bot, 1\}$,

for any $\phi \in LTL_+$ and $q \in Q_1$.

**Proof** Assume that $M_3$ is a $(\alpha_1, \alpha_2)$-abstraction of $M_1$ (the other case can be obtained analogously). Clearly, $Q_3 = Q_2/\rho'$ and $Q_2 = Q_3/\rho'$, where $\rho'$ is the identity.

Notice that Theorem 2.17, Corollary 2.9, and Corollary 2.10 take place even if $Q_2 = Q_1/\rho$ and $M_2$ is not necessarily an $(\alpha_R, \alpha_L)$-abstraction of $M_1$.

To prove 1 we show that the hypothesis of Theorem 2.17 hold for $M_3$ and $M_2$ (in this order):

- notice that $R_3([q]_\rho, [q']_\rho) \in \{1, \bot\}$ iff there exists $q_1 \in [q]_\rho$ and $q'_1 \in [q']_\rho$ such that $R_1(q_1, q'_1) \in \{1, \bot\}$ iff $R_2([q]_\rho, [q']_\rho) \in \{1, \bot\}$. Consequently, for any path $\pi_3$ in $M_3$ there exists a path $\pi_2$ in $M_2$ such that $|\pi_2| = |\pi_3|$ and $\pi_2(i) = \pi_3(i)$, for any $0 \leq i < |\pi_3|$, and vice-versa. Consequently, 1 and 2 in Theorem 2.17 hold for $M_2$;
- now, if $R_2([q]_\rho, [q']_\rho) = \bot$, there exist $q_1 \in [q]_\rho$ and $q'_1 \in [q']_\rho$ such that $R_1(q_1, q'_1) = \bot$ and $R_1(q_2, q'_2) \in \{\bot, 0\}$, for any $q_2 \in [q]_\rho$ and $q'_2 \in [q']_\rho$. Hence, $R_3([q]_\rho, [q']_\rho) = \bot$, which proves 3 in Theorem 2.17;
- if $L_2([q]_\rho, p) = \bot$, there exists $q_1 \in [q]_\rho$ such that $L_1(q_1, p) = \bot$ and $L_1(q_2, p) \in \{\bot, 1\}$, for any $q_2 \in [q]_\rho$. Hence, $L_3([q]_\rho, p) = \bot$, which proves 4 in Theorem 2.17;
- finally, if $L_2([q]_\rho, p) \in \{\bot, 1\}$, $L_1(q_1, p) \in \{\bot, 1\}$, for any $q_1 \in [q]_\rho$. Hence, $L_3([q]_\rho, p) \in \{\bot, 1\}$, which proves 5 in Theorem 2.17.

Therefore, from Theorem 2.17 we obtain 1.

To prove 2 we show that the hypothesis of Corollary 2.10 hold for $M_2$ and $M_3$ (in this order):

- the first hypothesis of Corollary 2.10 follows from what we have proved above;

- from $R_3([q]_\rho, [q']_\rho) = 1$ it follows that $R_1(q_1, q_1') = 1$, for any $q_1 \in [q]_\rho$ and $q_1' \in [q']_\rho$, and by the definition of $R_2$ we obtain that $R_2([q]_\rho, [q']_\rho) = 1$. Now, if $L_3([q]_\rho, p) = 1$, then $L_1(q_1, p) = 1$, for any $q_1 \in [q]_\rho$ and, consequently, $L_2([q]_\rho, p) = 1$.

Therefore, by Corollary 2.10 we obtain 2.

To prove 3 we show that the hypothesis of Corollary 2.9 hold for $M_2$ and $M_3$ (in this order):

- the first hypothesis of Corollary 2.9 follows from what we have proved above (in fact it is the same as the first hypothesis of Corollary 2.10);

- then, $M_2$ and $M_3$ have the same interpretation for the labeling function, which proves the second hypothesis of Corollary 2.9.

Therefore, we can apply Corollary 2.9 and we obtain 3. □

**Theorem 2.23** Let $M_1$ be an mv-Kripke structure over some set of atomic propositions and $\mathcal{B}_3$, $M_2$ an $(\alpha_2, \alpha_3)$- or an $(\alpha_2, \alpha_1)$-abstraction of $M_1$ based on some equivalence $\rho$, and $M_3$ an $(\alpha_1, \alpha_3)$- or a $(\alpha_1, \alpha_1)$-abstraction of $M_1$ based on $\rho$. Then,

$$[\phi]_{[q]}^{M_2} = 0 \Rightarrow [\phi]_{[q]}^{M_3} = 0,$$

for any $\phi \in LTL_+$ and $q \in Q_1$.

**Proof** Assume that $M_2$ is an $(\alpha_2, \alpha_3)$-abstraction and $M_3$ an $(\alpha_1, \alpha_3)$-abstraction of $M_1$ (the other cases can be obtained analogously). Clearly, $Q_2 = Q_3/\rho'$, where $\rho'$ is the identity.

We show that Corollary 2.11 can be applied to $M_3$ and $M_2$ (in this order). First, from $R_2([q]_\rho, [q']_\rho) \in \{1, \perp\}$ we obtain that $R_1(q_1, q_1') \in \{1, \perp\}$, for any $q_1 \in [q]_\rho$ and $q_1' \in [q']_\rho$, which implies that $R_3([q]_\rho, [q']_\rho) \in \{1, \perp\}$. Consequently, for any path $\pi_2$ in $M_2$ there exists a path $\pi_3$ in $M_3$ such that $\pi_2(i) = \pi_3(i)$, for any $0 \leq i < |\pi_2|$. Moreover, $|\pi_2| = |\pi_3|$.

If $L_2([q]_\rho, p) = 0$, then $L_1(q_1, p) = 0$, for any $q_1 \in [q]_\rho$. Hence, $L_3([q]_\rho, p) = 0$. □

**Theorem 2.24** Let $M_1$ be an mv-Kripke structure over some set of atomic propositions $AP$ and $\mathcal{B}_3$, $M_2$ an $(\alpha, \alpha')$-abstraction of $M_1$ based on some equivalence $\rho$, and $M_3$ an $(\alpha, \alpha'')$-abstraction of $M_1$ based on $\rho$, where $\alpha$, $\alpha'$, $\alpha'' \in \{\alpha_1, \alpha_2, \alpha_3\}$. If $\phi \in LTL_+$ and $q \in Q_1$ then:

1. If $\alpha' = \alpha_3$ and $\alpha'' = \alpha_1$ then:
   - $[\phi]^{M_3}_{[q]} = 1 \Rightarrow [\phi]^{M_2}_{[q]} = 1$;
   - $[\phi]^{M_2}_{[q]} = \bot \Rightarrow [\phi]^{M_3}_{[q]} = \bot$;
   - $[\phi]^{M_2}_{[q]} = 0 \Leftrightarrow [\phi]^{M_3}_{[q]} = 0$.
2. If $\alpha' = \alpha_2$ and $\alpha'' = \alpha_1$ then:
   - $[\phi]^{M_3}_{[q]} = 1 \Rightarrow [\phi]^{M_2}_{[q]} = 1$;
   - $[\phi]^{M_2}_{[q]} = \bot \Rightarrow [\phi]^{M_3}_{[q]} = \bot$;
   - $[\phi]^{M_2}_{[q]} = 1 \Leftrightarrow [\phi]^{M_3}_{[q]} = 1$.
3. If $\alpha' = \alpha_2$ and $\alpha'' = \alpha_3$ then:
   - $[\phi]^{M_2}_{[q]} = 1 \Rightarrow [\phi]^{M_3}_{[q]} = 1$;
   - $[\phi]^{M_3}_{[q]} = 0 \Rightarrow [\phi]^{M_2}_{[q]} = 0$;
   - $[\phi]^{M_2}_{[q]} = \bot \Leftrightarrow [\phi]^{M_3}_{[q]} \in \{\bot, 1\}$;
   - $[\phi]^{M_3}_{[q]} = \bot \Leftrightarrow [\phi]^{M_3}_{[q]} \in \{\bot, 0\}$.

**Proof** Directly from definitions, Theorem 2.17 and Corollaries 2.9, 2.10, and 2.11. □

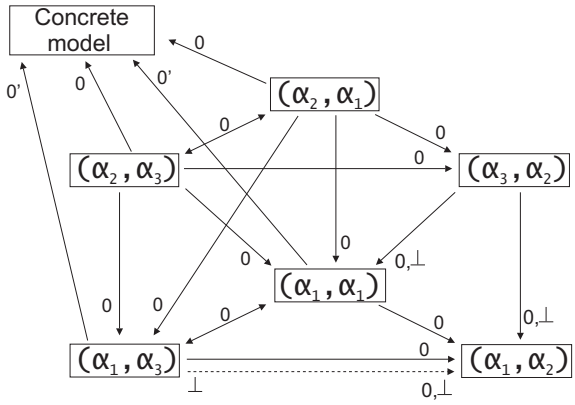


Figure 2.5: Preserving 0 truth values between different types of abstractions

The results developed in this section can be pictorially represented as in Figure 2.5 and Figure 2.6.

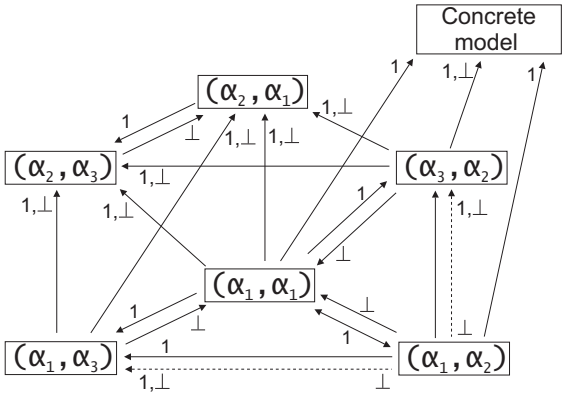


Figure 2.6: Preserving 1 and $\perp$ truth values between different types of abstractions

Both figures show preservation results for formulas in $LTL_+$: Figure 2.5 shows preservation results for the truth value 0, while the other figure shows preservation results for the truth values 1 and $\perp$. For example, an arrow from $(\alpha_2, \alpha_3)$ to $(\alpha_1, \alpha_1)$ labelled by 0 means that a property which is evaluated to 0 in a $(\alpha_2, \alpha_3)$-abstraction is also evaluated to 0 in a $(\alpha_1, \alpha_1)$-abstraction (both abstractions being under the same equivalence). An arrow like the one from $(\alpha_1, \alpha_2)$ to $(\alpha_1, \alpha_3)$ labelled by $\perp$ and $1, \perp$ (see Figure 2.6) means that a property which is evaluated to $\perp$ in a $(\alpha_1, \alpha_2)$-abstraction is evaluated to 1 or $\perp$ in a $(\alpha_1, \alpha_3)$-abstraction. Arrows labeled with $0'$ mean preservation of 0 values only for formulas in $LTL_+^\infty$.

These relationships offer some hints when choosing the type of abstraction to prove properties of systems. For example, when trying to falsify some property in $LTL_+^\infty$ we can choose to make a $(\alpha_2, \alpha_1)$-abstraction or a $(\alpha_1, \alpha_1)$-abstraction of the system. Both abstractions preserve the truth value 0 from the abstract system to the concrete one. However, the former removes more behavior of the system and, consequently, it is easier to check while the latter preserves more behavior and can be used to falsify more properties.

## 2.4 Temporal Logic of Knowledge

### 2.4.1 Abstractions and Preservation Results

In this section, we extend the abstractions of mv-Kripke structures from the previous section to the multi-agent case [45]. As we have seen, the transition predicate and the labeling function of the abstract system are obtained by reinterpretations according to some interpretation policy. In the case of the

similarity relations of the abstract system, we can not always use the reinterpretations of the similarity relations of the concrete system according to some interpretation policy. These reinterpretations might not satisfy reflexivity, symmetry or transitivity. We will discuss the reinterpretations used in [45] which correspond to the following safe interpretation policies on the truth algebra corresponding to Kleene's 3-valued interpretation, denoted $\mathcal{B}_3$:

- $\alpha_1(0) = \forall$, $\alpha_1(\perp) = \exists_a^{\{0,\perp,1\}}$ and $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \exists^{\{0,\perp,1\}}$, $\alpha_2(\perp) = \exists^{\{\perp,1\}}$ and $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\perp) = \exists^{\{0,\perp\}}$ and $\alpha_3(1) = \exists^{\{0,\perp,1\}}$.

The relations between the reinterpretations according to these interpretation policies and the properties of a similarity relation are enumerated in the next proposition.

**Proposition 2.1** Let $M = (Q, R, L, (\sim_i |\ 1 \leq i \leq n))$ be a multi-agent multi-valued Kripke structure over $AP$ and $\mathcal{B}_3$, and $\rho$ an equivalence relation on $Q$. We have that:

1. the reinterpretation of $\sim_i$ according to $\alpha_1$ is symmetric;
2. the reinterpretation of $\sim_i$ according to $\alpha_2$ is symmetric and transitive;
3. the reinterpretation of $\sim_i$ according to $\alpha_3$ is reflexive and symmetric.

**Proof** Clearly, the symmetry of $\sim_i$ implies the symmetry of the reinterpretation of $\sim_i$ according to any interpretation policy. Moreover, $\alpha_3(1) = \exists^{\{0,\perp,1\}}$ implies that the reinterpretation of $\sim_i$ according to $\alpha_3$ is reflexive.

Now, let $\delta : Q/\rho \times Q/\rho \to \{0, 1, \perp\}$ be the reinterpretation of $\sim_i$ according to $\alpha_2$. To prove the transitivity of $\delta$, the following cases are to be discussed:

1. $\delta([q],[q']) = 0$ and $\delta([q'],[q'']) = \perp$, for some $q, q', q'' \in Q_1$. By the definition of $\delta$, there exist $q_1 \in [q]$, $q_1' \in [q']$ such that $\sim_i (q_1, q_1') = 0$, and $\sim_i (q_2', q_1'') \in \{\perp, 1\}$, for any $q_2' \in [q']$ and $q_1'' \in [q'']$.

   Since $\sim_i$ is a similarity relation, we have that $\sim_i (q_1, q_1'') = 0$, for any $q_1'' \in [q'']$. Hence, $\delta([q],[q'']) = 0$.

2. $\delta([q],[q']) = \perp$ and $\delta([q'],[q'']) = 0$, for some $q, q', q'' \in Q_1$. This case is similar to the previous one.

3. $\delta([q],[q']) = 0$ and $\delta([q'],[q'']) = 1$, for some $q, q', q'' \in Q_1$. By the definition of $\delta$, there exist $q_1 \in [q]$ and $q_1' \in [q']$ such that $\sim_i (q_1, q_1') = 0$, and $\sim_i (q_2', q_1'') = 1$, for any $q_2' \in [q']$ and $q_1'' \in [q'']$.

   Again, because $\sim_i$ is a similarity relation, we obtain $\sim_i (q_1, q_1'') = 0$, for any $q_1'' \in [q'']$. Consequently, $\delta([q],[q'']) = 0$.

4. $\delta([q],[q']) = 1$ and $\delta([q'],[q'']) = 0$, for some $q, q', q'' \in Q_1$. This case is similar to the previous one.

5. $\delta([q],[q']) = \perp$ and $\delta([q'],[q'']) = 1$, for some $q, q', q'' \in Q_1$. The former implies that there exist $q_1 \in [q]$ and $q_1' \in [q']$ such that $\sim_i (q_1, q_1') = \perp$, while the latter implies that $\sim_i (q_2', q_1'') = 1$, for any $q_2' \in [q']$ and $q_1'' \in [q'']$.

   By the transitivity of $\sim_i$, $\sim_i (q_1, q_1'') = \perp$, for any $q_1'' \in [q'']$. Moreover, by the definition of $\delta$, $\sim_i (q_2, q_3') \in \{\perp, 1\}$, for any $q_2 \in [q]$ and $q_3' \in [q']$, which implies $\sim_i (q_2, q_1'') \in \{\perp, 1\}$, for any $q_2 \in [q]$ and $q_1'' \in [q'']$. Hence, $\delta([q],[q'']) = \perp$.

6. $\delta([q],[q']) = 1$ and $\delta([q'],[q'']) = \perp$, for some $q, q', q'' \in Q_1$. This case is similar to the previous one.

7. $\delta([q],[q']) = \perp$ and $\delta([q'],[q'']) = \perp$, for some $q, q', q'' \in Q_1$. The first property implies that there exist $q_1 \in [q]$ and $q_1' \in [q']$ such that $\sim_i (q_1, q_1') = \perp$ and the second property implies that $\sim_i (q_2', q_1'') \in \{\perp, 1\}$, for any $q_2' \in [q']$ and $q_1'' \in [q'']$. By the transitivity of $\sim_i$, we obtain that $\sim_i (q_1, q_1'') = \perp$, for any $q_1'' \in [q'']$. Moreover, we have that $\sim_i (q_2, q_3') \in \{\perp, 1\}$, for any $q_2 \in [q]$ and $q_3' \in [q']$, which implies $\sim_i (q_2, q_1'') \in \{\perp, 1\}$, for any $q_2 \in [q]$ and $q_1'' \in [q'']$. Hence, $\delta([q],[q'']) = \perp$.

8. $\delta([q],[q']) = 1$ and $\delta([q'],[q'']) = 1$, for some $q, q', q'' \in Q_1$. From the former we obtain that $\sim_i (q_1, q_1') = 1$, for any $q_1 \in [q]$ and $q_1' \in [q']$ and from the latter $\sim_i (q_2', q_1'') = 1$, for any $q_2' \in [q']$ and $q_1'' \in [q'']$. Since $\sim_i$ is transitive, we get that $\sim_i (q_1, q_1'') = 1$ for any $q_1 \in [q]$ and $q_1'' \in [q'']$. Consequently, $\delta([q],[q'']) = 1$. □

The above proposition mentions all the properties of a similarity relation that are always satisfied by a reinterpretation of $\sim_i$ according to $\alpha_1$, $\alpha_2$ or $\alpha_3$. Consequently, when trying to redefine the similarity relations in the abstract system we have to apply some reflexive or transitive closures.

Before we define closures, we remark that the properties of a similarity relation $\sim$: $A \times A \to \{0, \perp, 1\}$ build classes of elements that have the same

properties as the equivalence classes build by some equivalence relation. The class of an element $a \in A$ is

$$[a]_{\sim} = \{x | \sim_i (a, x) \neq 0\}.$$

The class $[a]_{\sim}$ has the property that $\sim (x, y) = \sim (x', y')$ for any $x, y, x', y' \in [a]_{\sim}$ with $x \neq y$ and $x' \neq y'$. To prove this, let $\sim (x, y) = b_1$ and $\sim (y, x') = b_2$. By the transitivity of $\sim$, we obtain that $\sim (x, x') = b_1 \wedge b_2$. The symmetry of $\sim$ implies $\sim (x', y) = b_2$ which, by $\sim (x, y) = \sim (x, x') \wedge \sim (x', y)$, further implies $b_1 = b_1 \wedge b_2$ and consequently, $b_1 \leq b_2$. In a similar manner, we can prove $b_2 \leq b_1$ which concludes $b_1 = b_2$. Now, using a similar approach, we can prove $\sim (y, x') = \sim (x', y')$ which completes our proof.

Let $\sigma : A \times A \to \{0, 1, \perp\}$ be a three-valued relation. The *reflexive closure of* $\sigma$ is a three-valued relation $\sigma_r : A \times A \to \{0, 1, \perp\}$ defined by:

$$\sigma_r(x, y) = \begin{cases} 1, & \text{if } x = y; \\ \sigma(x, y), & \text{otherwise.} \end{cases}$$

The transitive closure of a reflexive and symmetric three-valued relation $\sigma : A \times A \to \{0, 1, \perp\}$ is build as follows. For each $a \in A$, we construct the class $[a]_{\sigma}$ by computing the sequence of sets $([a]^i_{\sigma} | \ i \geq 0)$ defined by:

$$\begin{aligned} [a]^0_{\sigma} &= \{a\}; \\ [a]^{i+1}_{\sigma} &= \{x \mid (\exists y \in [a]^i_{\sigma})(\sigma(x, y) \neq 0)\}, \end{aligned}$$

until $[a]^j_{\sigma} = [a]^{j+1}_{\sigma}$, for some $j \geq 0$. We take $[a]_{\sigma} = [a]^j_{\sigma}$ and define the transitive closure $\sigma_t : A \times A \to \{0, 1, \perp\}$ as follows:

- if a class $[a]_{\sigma}$ has the property $\sigma(x, y) \neq 0$, for all $x, y \in [a]_{\sigma}$, then we take $\sigma_t(x, y) = \sigma(x, y)$, for all $x, y \in [a]_{\sigma}$;
- otherwise, we take $\sigma_t(x, x) = 1$, for all $x \in [a]_{\sigma}$, and $\sigma_t(x, y) = \perp$, for all $x, y \in [a]_{\sigma}$ with $x \neq y$;
- for any $a_1, a_2 \in A$ with $[a_1]_{\sigma} \cap [a_2]_{\sigma} = \emptyset$, we take $\sigma_t(x, y) = 0$, for all $x \in [a_1]_{\sigma}$ and $y \in [a_2]_{\sigma}$.

**Remark 2.10** We may remark that the transitive closure of $\sigma$, $\sigma_t$, transforms 0 and 1 values of $\sigma$ into $\perp$ values.

**Definition 2.22** Let $M = (Q, R, L, (\sim_i | \ 1 \leq i \leq n))$ be a multi-agent multi-valued Kripke structure over $AP$ and $\mathcal{B}_3$, $\rho$ an equivalence relation on $Q$, and $\alpha_R$, $\alpha_L$, $\alpha_S \in \{\alpha_1, \alpha_2, \alpha_3\}$ three interpretation policies over $\mathcal{B}_3$. A multi-agent mv-Kripke structure $M' = (Q', R', L', (\sim'_i | \ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$ is called an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction of $M$ by $\rho$ if:

- $Q' = Q/_\rho$;
- $R'$ is the reinterpretation of $R$ on $Q' \times Q'$ according to $\alpha_R$;
- $L'$ is the reinterpretation of $L$ over $Q'$ according to $\alpha_L$.
- $\sim_i'$ is defined as follows:
  - if $\alpha_S = \alpha_1$ then $\sim_i'$ is the reflexive and transitive closure of the reinterpretation of $\sim_i$ on $Q' \times Q'$ according to $\alpha_1$;
  - if $\alpha_S = \alpha_2$ then $\sim_i'$ is the reflexive closure of the reinterpretation of $\sim_i$ on $Q' \times Q'$ according to $\alpha_2$;
  - if $\alpha_S = \alpha_3$ then $\sim_i'$ is the transitive closure of the reinterpretation of $\sim_i$ on $Q' \times Q'$ according to $\alpha_3$.

**Example 2.12** Consider the system from Figure 2.7. It consists of two agents $H$ and $L$ asynchronously composed (the $\wedge$ symbol means that the instructions are executed in one atomic transition). $H$ continuously increases $x$ by 2 but, when it receives a signal from $L$ ($L$ sets $z$ to 1) it increases $x$ only by 1. The agent $L$ also uses $y$ to count the number of signals it sends.

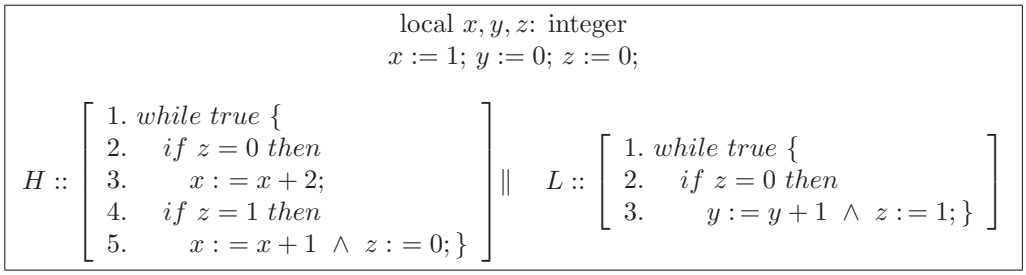


Figure 2.7: A system with two processes

Suppose that $L$ can not read $x$. We would be interested in checking whether the low-level user $L$ can deduce something about the parity of $x$.

We model this system by a multi-agent mv-Kripke structure

$$M_1 = (Q, R, L, (\sim_H, \sim_L)),$$

whose states are triples $(x, y, z)$, where $x$, $y$ and $z$ are the variables from Figure 2.7. We consider that $(x, y, z) \sim_H (x', y', z')$ if $x = x'$ and $z = z'$ and $(x, y, z) \sim_L (x', y', z')$ if $y = y'$ and $z = z'$.

An instance of the question above is the formula

$$\phi = P_L \ \mathbf{O} \ \big((even(y) \wedge z = 1) \Rightarrow \mathbf{P} \ \neg(even(x) \ \mathbf{S} \ z = 1)\big),$$

where *even* is a predicate that it is 1 for even numbers and 0 otherwise. If this property is false then, the fact that $y$ is even implies that $x$ is even in all the states since the last time $z$ was 1. Consequently, if $y$ is even $L$ knows the parity of $x$ for some of the past states.

Such a property is an instance of information flow w.r.t *run-based secrecy* [97]. In fact, the falsity of $\phi$ would prove also that agent $H$ does not maintain run-based secrecy with respect to agent $L$, i.e. there exists an $H$-local and satisfiable formula $\phi$ such that $P_L\ \mathbf{O}\phi$ is false.

We define the equivalence relation $\rho$ on the set of states of $M_1$ as follows:

$$(x, y, z)\ \rho\ (x', y', z') \text{ if } even(|x - x'|) \wedge even(|y - y'|) \wedge z = z'.$$

The $(\alpha_3, \alpha_3, \alpha_3)$-abstraction of $M_1$ by $\rho$ is depicted in Figure 2.8(a). We have denoted the abstract states by triples, where the first two elements specify the parity of $x$ and $y$, respectively, while the third component gives the value of $z$.

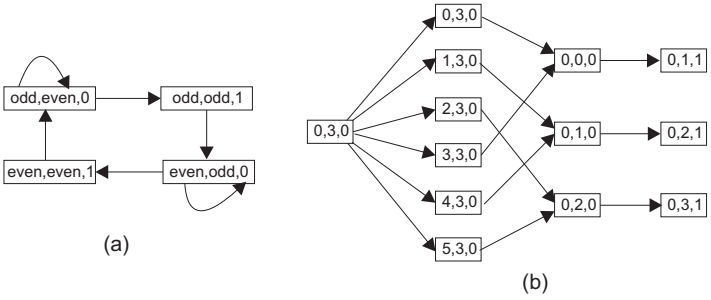


Figure 2.8: (a) An $(\alpha_3, \alpha_3, \alpha_3)$-abstraction of $M_1$. (b) An $(\alpha_2, \alpha_2, \alpha_2)$-abstraction of $M_1$.

We can remark that the abstract system has only 4 reachable states and it is very easy to verify it. All we need is some preservation results in order to transfer the truth value of $\phi$ from the abstract system into the concrete system.

**Example 2.13** The system in Figure 2.9 consists of two agents $H$ and $L$ (as before, $\wedge$ means that the instructions linked by it are executed in one atomic transition) which run in parallel. $H$ generates randomly a value for $x$ and then, according to the remainder $x$ mod 6, it sets $y$. When $H$ have set $y$, $L$ increments it and stops the system by setting $z = 1$.

We suppose that $x$ is hidden to $L$ and we want to answer the following question: can $L$ deduce something about the remainder $x\ mod\ 6$ from the parity of $y$?

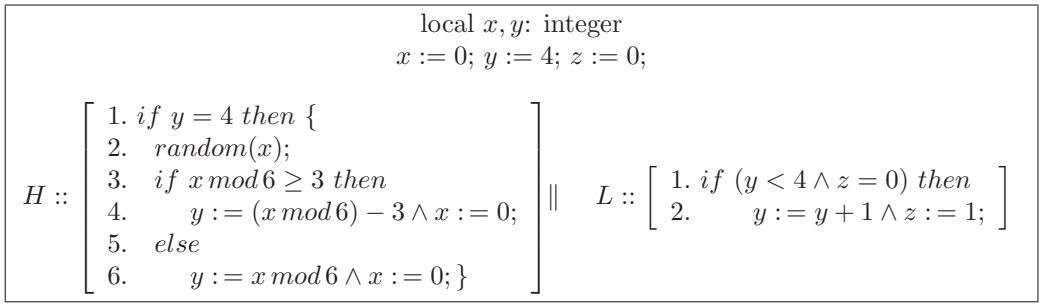


Figure 2.9: A system with two processes

The system is modeled by a multi-agent mv-Kripke structure

$$M_2 = (Q', R', L', (\sim'_H, \sim'_L)),$$

whose states are triples $(x, y, z)$, where $x$, $y$ and $z$ are the variables from Figure 2.9. We consider $(x, y, z) \sim'_H (x', y', z')$ if $x = x'$ and $(x, y, z) \sim'_L (x', y', z')$ if $even(|y - y'|)$.

The question above can be expressed by the following formulas:

$$\phi_k = P_L\ \mathbf{O}\ \big((z = 0) \vee \mathbf{O}\ (x\ mod\ 6 = k)\big),\ \text{for any } 0 \leq k \leq 5.$$

If all these formulas are true then we obtain a negative answer to our question. Again, we are trying to prove properties that are related to the notion of run-based secrecy from [97].

Now, consider the equivalence relation $\sigma$ on $Q'$ given by:

$$(x, y, z)\ \sigma\ (x', y', z')\ \text{if}\ x\ mod\ 6 = x'\ mod\ 6 \wedge y = y' \wedge z = z'.$$

The $(\alpha_2, \alpha_2, \alpha_2)$-abstraction of $M'$ by $\sigma$ is shown in Figure 2.8(b). We have denoted abstract states by triples $(a, b, c)$, where $a = x\ mod\ 6$, $b = y$ and $c = z$. Again, we have obtained a finite abstract system and it will be very useful if we could verify properties directly on it.

Abstractions are useful when they offer preservation results with respect to a specific set of properties. We offer three forms of property preserving, the first two being frequently found in the literature [34]:

- *weak-preservation with respect to a set of properties $P$*: an abstraction is weakly preserving with respect to $P$ if for any $\phi \in P$, $\phi$ is evaluated to 1 in the abstract system implies that $\phi$ is evaluated to 1 in the concrete system;

- *error-preservation with respect to a set of properties $P$*: an abstraction is error preserving with respect to $P$ if for any $\phi \in P$, $\phi$ is evaluated to 0 in the abstract system implies that $\phi$ is evaluated to 0 in the concrete system;

- *very weak-preservation with respect to a set of properties $P$*: an abstraction is very weak-preserving with respect to $P$ if for any $\phi \in P$, $\phi$ is evaluated to 1 or $\perp$ in the abstract system implies that $\phi$ is evaluated to 1 or $\perp$ in the concrete system.

### Weak and very weak preservation results

In order to prove weak and very weak preservation results, we start by some technical lemmas that will identify the basic conditions that must be satisfied by the abstraction in order to be weak or very weak preserving with respect to $\forall KCTL^*P_+$ or $\exists KCTL^*P_+$ formulas.

Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$. As in the case of mv-Kripke structures, we say that a path $\pi_2 \in \mathsf{Paths}(M_2, \{\perp, 1\})$ is a *corresponding path* to $\pi_1 \in \mathsf{Paths}(M_1, \{\perp, 1\})$ if:

- $|\pi_2| = |\pi_1|$;

- $\pi_2(i) = [\pi_1(i)]$, for any $0 \leq i < |\pi_2|$.

We denote by $C_{M_1}(\pi_2)$ the set of all $\{\perp, 1\}$-paths in $M_1$ that have $\pi_2$ as a corresponding $\{\perp, 1\}$-path in $M_2$.

**Lemma 2.4** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ a $\forall KCTL^*P_+$ formula. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, \{\perp, 1\})$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, \{\perp, 1\})$;

2. for any $q, q' \in Q_1$, $\sim_i^1 (q, q') \in \{\perp, 1\}$ implies $\sim_i^2 ([q], [q']) \in \{\perp, 1\}$;

3. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \in \{\perp, 1\}$ implies $L_1(q)(p) \in \{\perp, 1\}$,

then

$$([\phi]^{I_2}_{(\pi_2,m)} \in \{\bot, 1\}) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively).

**Proof** The claim can be proved by structural induction on $\phi$. We will present only some of the possible cases, the others being similar to these ones:

- $\phi = p \in AP$. Suppose that $[\phi]^{I_2}_{(\pi_2,m)} = L_2(\pi_2(m), p) \in \{\bot, 1\}$, for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. By 3 and $\pi_2(m) = [\pi_1(m)]$, we obtain that $L_1(\pi_1(m), p) \in \{\bot, 1\}$ and, therefore, $[\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;

- $\phi = \overline{\mathbf{X}}\phi_1$. Assume that $\phi_1$ satisfies the property and

  $$[\overline{\mathbf{X}}\phi_1]^{I_2}_{(\pi_2,m)} = |\pi_2^m| \leq 1 \vee \big([\phi_1]^{I_2}_{(\pi_2,m+1)} \wedge R_2(\pi_2(m), \pi_2(m+1))\big) \in \{\bot, 1\},$$

  for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. If $|\pi_2^m| \leq 1$ then $|\pi_1^m| \leq 1$ which leads to $[\overline{\mathbf{X}}\phi_1]^{I_1}_{(\pi_1,m)} = 1$. Otherwise, $[\phi_1]^{I_2}_{(\pi_2,m+1)} \wedge R_2(\pi_2(m), \pi_2(m+1)) \in \{\bot, 1\}$. By the definition of $\pi_1$ we obtain $|\pi_1^m| > 1$ and $R_1(\pi_1(m), \pi_1(m+1)) \in \{\bot, 1\}$, and by the induction hypothesis, $[\phi_1]^{I_1}_{(\pi_1,m+1)} \in \{\bot, 1\}$. Consequently, $[\overline{\mathbf{X}}\phi_1]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;

- $\phi = \forall\phi_1$. Assume that $\phi_1$ satisfies the property and

  $$[\forall\phi_1]^{I_2}_{(\pi_2,m)} = \bigwedge_{\pi[1..m]=\pi_2[1..m]} [\phi_1]^{I_2}_{(\pi,m)} \in \{\bot, 1\},$$

  for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. This implies $[\phi_1]^{I_2}_{(\pi,m)} \in \{\bot, 1\}$, for all paths $\pi$ such that $\pi[1..m] = \pi_2[1..m]$. By 1, for each path $\sigma$ of $M_1$ with $\sigma[1..m] = \pi_1[1..m]$ there exists a path $\sigma'$ of $M_2$ such that $\sigma'[1..m] = \pi_2[1..m]$ and $[\phi_1]^{I_2}_{(\sigma',m)} \in \{\bot, 1\}$. Applying the induction hypothesis we get $[\phi_1]^{I_1}_{(\sigma,m)} \in \{\bot, 1\}$, for each path $\sigma$ with $\sigma[1..m] = \pi_1[1..m]$. Hence, $[\forall\phi_1]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;

- $\phi = \mathbf{K}_i\phi_1$. Assume that $\phi_1$ satisfies the property and

  $$\bigwedge_{\sim_i^2((\pi_2,m),(\pi_2'',m'))\neq 0} \sim_i^2 ((\pi_2, m), (\pi_2'', m')) \wedge [\phi]^{I_2}_{(\pi_2'',m')} \in \{\bot, 1\}, \quad (2.1)$$

for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. We have to prove that

$$\bigwedge_{\sim_i^1((\pi_1,m),(\pi_1',m''))\neq 0} \sim_i^1 ((\pi_1, m), (\pi_1', m'')) \wedge [\phi]^{I_1}_{(\pi_1',m'')} \in \{\bot, 1\}.$$

Let $(\pi_1', m'')$ be a point of $M_1$ such that $\sim_i^1 ((\pi_1, m), (\pi_1', m'')) \neq 0$. By 1, there exists a corresponding path for $\pi_1'$, denote it $\pi_2'$, such that $\pi_2'(m'') = [\pi_1'(m'')]$. Moreover, by 2, $\sim_i^2 ((\pi_2, m), (\pi_2', m'')) \in \{\bot, 1\}$. From property (2.1) we deduce that $[\phi_1]^{I_2}_{(\pi_2',m'')} \in \{\bot, 1\}$ which, by the induction hypothesis, implies $[\phi_1]^{I_1}_{(\pi_1',m'')} \in \{\bot, 1\}$. □

**Lemma 2.5** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ a $\forall KCTL^*P_+$ formula. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q, q' \in Q_1$, $\sim_i^1 (q, q') \in \{\bot, 1\}$ implies $\sim_i^2 ([q], [q']) \in \{\bot, 1\}$;
3. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 1$ implies $L_1(q)(p) = 1$;
4. for any $q, q' \in Q_1$, $R_2([q], [q']) = 1$ implies $R_1(q, q') = 1$;
5. for any $q, q' \in Q_1$, $\sim_i^2 ([q], [q']) = 1$ and $[q] \neq [q']$ implies $\sim_i^1 (q, q') = 1$,

then

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively). The same holds even if the requirements 4 and 5 are replaced by $R_1(q, q') \in \{0, 1\}$, for any $q, q' \in Q_1$, and $\sim_i^1 (q, q') \in \{0, 1\}$, for any $q, q' \in Q_1$, respectively.

**Proof** The proof follows the same lines as Lemma 2.4. We detail only the following case:

- $\phi = \mathbf{K}_i\phi_1$. Assume that $\phi_1$ satisfies the property and

$$\bigwedge_{\sim_i^2((\pi_2,m),(\pi_2'',m'))\neq 0} \sim_i^2 ((\pi_2, m), (\pi_2'', m')) \wedge [\phi]^{I_2}_{(\pi_2'',m')} = 1, \qquad (2.2)$$

for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. We have to prove that

$$\bigwedge_{\sim_i^1((\pi_1,m),(\pi_1',m''))\neq 0} \sim_i^1 ((\pi_1, m), (\pi_1', m'')) \wedge [\phi]^{I_1}_{(\pi_1', m'')} = 1.$$

Let $(\pi_1', m'')$ be a point of $M_1$ such that $\sim_i^1 ((\pi_1, m), (\pi_1', m'')) \neq 0$.

If $(\pi_1', m'') = (\pi_1, m)$ then $\sim_i^1 ((\pi_1, m), (\pi_1, m)) = 1$. Because $\sim_i^2$ is a similarity relation, we have $\sim_i^2 ((\pi_2, m), (\pi_2, m)) = 1$ which implies $[\phi]^{I_2}_{(\pi_2,m)} = 1$. By the induction hypothesis, we obtain $[\phi]^{I_1}_{(\pi_1,m)} = 1$.

If $(\pi_1', m'') \neq (\pi_1, m)$, by 1, we obtain that there exists a corresponding path for $\pi_1'$, denote it $\pi_2'$, such that $\pi_2'(m'') = [\pi_1'(m'')]$. Moreover, by 2, $\sim_i^2 ((\pi_2, m), (\pi_2', m'')) \in \{\bot, 1\}$. From the property (2.2), we obtain that $\sim_i^2 ((\pi_2, m), (\pi_2', m'')) = 1$ and $[\phi_1]^{I_2}_{(\pi_2', m'')} = 1$ and finally, by 5 and the induction hypothesis, we get $\sim_i^1 ((\pi_1, m), (\pi_1', m'')) = 1$ and $[\phi_1]^{I_1}_{(\pi_1', m'')} = 1$, which completes our proof. □

Until now, we have proved weak and very weak preservation results involving $\forall KCTL^*P_+$ formulas. Now, we turn our attention to $\exists KCTL^*P_+$ formulas.

**Lemma 2.6** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ an $\exists KCTL^*P_+$ formula. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, \{\bot, 1\})$ there exists a path $\pi_1 \in \mathsf{Paths}(M_1, \{\bot, 1\})$ with $\pi_1 \in C_{M_1}(\pi_2)$;
2. for any $q, q' \in Q_1$, $\sim_i^2 ([q], [q']) \in \{\bot, 1\}$ implies $\sim_i^1 (q, q') \in \{\bot, 1\}$;
3. for any $q, q' \in Q_1$, $R_2([q], [q']) \in \{\bot, 1\}$ implies $R_1(q, q') \in \{\bot, 1\}$;
4. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) \in \{\bot, 1\}$ implies $L_1(q)(p) \in \{\bot, 1\}$,

then

$$([\phi]^{I_2}_{(\pi_2,m)} \in \{\bot, 1\}) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively).

**Proof** We can proceed by structural induction on $\phi$ as in Lemma 2.4. Below, we give only the new cases.

- $\phi = \exists\phi_1$. Assume that $\phi_1$ satisfies the property and let $(\pi_2, m)$ be a point of $M_2$ such that

$$[\exists\phi_1]^{I_2}_{(\pi_2,m)} = \bigvee_{\pi'_2[1..m]=\pi_2[1..m]} [\phi_1]^{I_2}_{(\pi'_2,m)} \in \{\bot, 1\}.$$

  There exists a path $\pi'_2$ with $\pi'_2[1..m] = \pi_2[1..m]$ such that $[\phi_1]^{I_2}_{(\pi'_2,m)} \in \{\bot, 1\}$. Let $\pi_1$ be a path of $M_1$ corresponding to $\pi_2$ (by 1, we know that it exists). Applying the induction hypothesis, we get that $[\phi_1]^{I_1}_{(\pi'_1,m)} \in \{\bot, 1\}$, for any path $\pi'_1 \in C_{M_1}(\pi'_2)$. Because of the third condition and $\pi'_2[1..m] = \pi_2[1..m]$, there exists $\pi''_1 \in C_{M_1}(\pi'_2)$ such that $\pi''_1[1..m] = \pi_1[1..m]$ and $[\phi_1]^{I_1}_{(\pi''_1,m)} \in \{\bot, 1\}$ which implies $[\exists\phi_1]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;

- $\phi = \mathbf{P}_i\phi_1$. Assume that $\phi_1$ satisfies the property and let $(\pi_2, m)$ be a point of $M_2$ such that

$$\bigvee_{\sim^2_i((\pi_2,m),(\pi'_2,m'))\neq 0} \sim^2_i ((\pi_2, m), (\pi'_2, m')) \wedge [\phi]^{I_2}_{(\pi'_2,m')} \in \{\bot, 1\}.$$

  There exists $(\pi'_2, m')$ a point of $M_2$ such that $\sim^2_i ((\pi_2, m), (\pi'_2, m')) \in \{\bot, 1\}$ and $[\phi]^{I_2}_{(\pi'_2,m')} \in \{\bot, 1\}$. Now, let $\pi_1 \in C_{M_1}(\pi_2)$ and $\pi'_1 \in C_{M_1}(\pi'_2)$ (by 1, we know that they exist). By 2, we have $\sim^1_i ((\pi_1, m), (\pi'_1, m')) \in \{\bot, 1\}$. Moreover, by the induction hypothesis, we obtain $[\phi_1]^{I_1}_{(\pi'_1,m')} \in \{\bot, 1\}$ and consequently,

$$[\mathbf{P}_i\phi_1]^{I_1}_{(\pi_1,m)} = \bigvee_{\sim^1_i((\pi_1,m),(\pi'_1,m'))\neq 0} \sim^1_i ((\pi_1, m), (\pi'_1, m'))\wedge[\phi]^{I_1}_{(\pi'_1,m')} \in \{\bot, 1\}.$$

□

**Lemma 2.7** Let $M_2 = (Q_2, R_2, L_2, (\sim^2_i |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim^1_i |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ an $\exists KCTL^*P_+$ formula. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, \{\bot, 1\})$ there exists a path $\pi_1 \in \mathsf{Paths}(M_1, \{\bot, 1\})$ with $\pi_1 \in C_{M_1}(\pi_2)$;

2. for any $q, q' \in Q_1$, $\sim_i^2 ([q], [q']) = 1$ implies $\sim_i^1 (q, q') = 1$;

3. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 1$ implies $L_1(q)(p) = 1$;

4. for any $q, q' \in Q_1$, $R_2([q], [q']) = 1$ implies $R_1(q, q') = 1$,

then

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively).

**Proof** The proof follows the same lines as Lemma 2.6. □

We will enumerate now the weak and very weak preservation results that hold for all the possible types of abstraction.

**Theorem 2.25** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 | \ 1 \leq i \leq n))$ be an $(\alpha_3, \alpha_L, \alpha_3)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 | \ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ an $\forall KCTL^*P_+$ formula.

- If $\alpha_L = \alpha_2$ then

  $$([\phi]^{I_2}_{(\pi_2,m)} \in \{\bot, 1\}) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Moreover, if $R_1(q, q') \in \{0, 1\}$ and $\sim_i^1 (q, q') \in \{0, 1\}$, for all $q, q' \in Q_1$ and $1 \leq i \leq n$, then

  $$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

- If $\alpha_L = \alpha_1$, $R_1(q, q') \in \{0, 1\}$ and $\sim_i^1 (q, q') \in \{0, 1\}$, for all $q, q' \in Q_1$ and $1 \leq i \leq n$, then

  $$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemma 2.4 and Lemma 2.5. □

**Theorem 2.26** Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 | \ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure over $AP$ and $\mathcal{B}_3$, $M_2 = (Q_2, R_2, L_2, (\sim_i^2 | \ 1 \leq i \leq n))$ an $(\alpha_1, \alpha_L, \alpha_1)$-abstraction by an equivalence $\rho$, and $\phi$ an $\forall KCTL^*P_+$ formula.

- If $\alpha_L \in \{\alpha_1, \alpha_2\}$ then

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemma 2.5. □

Next, we try to give a weak and a very weak preservation result for abstractions of type $(\alpha_2, \alpha_L, \alpha_2)$, for some $\alpha_L \in \{\alpha_1, \alpha_2, \alpha_3\}$, with respect to $\exists KCTL^*P_+$ formulas. Since the similarity relations of the abstractions are reflexive closures of the reinterpretations of $\sim_i^1$ according to $\alpha_2$, the condition 2 in Lemmas 2.6 and 2.7 might not hold. However, we will show that the preservation results hold for equivalences relations $\rho$ for which the reinterpretation of $\sim_i^1$ according to $\alpha_2$ is already reflexive. If $M_1$ and $M_2$ are as above, we say that $\rho$ *is compatible with* $\sim_i^1$ if whenever $q\ \rho\ q'$ we have $\sim_i^1 (q, q') = 1$. Clearly, this compatibility implies that the reinterpretation of $\sim_i^1$ according to $\alpha_2$ is reflexive.

**Theorem 2.27** Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure over $AP$ and $\mathcal{B}_3$, $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ an $(\alpha_2, \alpha_L, \alpha_2)$-abstraction by an equivalence $\rho$ compatible with $\sim_i^1$, and $\phi$ an $\exists KCTL^*P_+$ formula.

- If $\alpha_L \in \{\alpha_1, \alpha_2\}$ then

$$([\phi]^{I_2}_{(\pi_2,m)} = 1) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 1),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

- If $\alpha_L = \alpha_2$ then

$$([\phi]^{I_2}_{(\pi_2,m)} = \bot) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}),$$

  for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemmas 2.6 and 2.7. □

**Example 2.14** The abstract system from Example 2.13 has only 13 reachable states and we can easily prove that the truth value of $\phi_k$, for any $0 \leq k \leq 5$, is 1. Moreover, since $\phi_k \in \exists KCTL^*P_+$ we can apply the weak preservation result from Theorem 2.27 and obtain that all formulas $\phi_k$ are 1 in the concrete system $I_1$.

**Error preservation results**

To prove the error preservation results, we start again, by some technical lemmas that identify the basic conditions that must be satisfied by an abstraction in order to be error-preserving.

**Lemma 2.8** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ an $\exists KCTL^*P_+$ formula. If

1. for any $\pi_1 \in \mathsf{Paths}(M_1, D)$ there exists a corresponding path $\pi_2 \in \mathsf{Paths}(M_2, D)$;
2. for any $q, q' \in Q_1$, $\sim_i^1 (q, q') \in \{\bot, 1\}$ implies $\sim_i^2 ([q], [q']) \in \{\bot, 1\}$;
3. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 0$ implies $L_1(q)(p) = 0$;

then

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively).

**Proof** The claim is proved by structural induction on $\phi$. We will enumerate only some of the possible cases:

- $\phi = \exists \phi_1$. Assume that $\phi_1$ satisfies the property and

  $$[\exists \phi_1]^{I_2}_{(\pi_2,m)} = \bigvee_{\pi[1..m]=\pi_2[1..m]} [\phi_1]^{I_2}_{(\pi,m)} = 0,$$

  for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists. Then, $[\phi_1]^{I_2}_{(\pi,m)} = 0$, for all paths $\pi$ such that $\pi[1..m] = \pi_2[1..m]$.

  By 1, for each path $\sigma$ of $M_1$ with $\sigma[1..m] = \pi_1[1..m]$ there exists a path $\sigma'$ of $M_2$ such that $\sigma'[1..m] = \pi_2[1..m]$ and $[\phi_1]^{I_2}_{(\sigma',m)} = 0$. Applying the induction hypothesis, we get $[\phi_1]^{I_1}_{(\sigma,m)} = 0$, for each path $\sigma$ with $\sigma[1..m] = \pi_1[1..m]$. Hence, $[\exists \phi_1]^{I_1}_{(\pi_1,m)} = 0$

- $\phi = \mathbf{P}_i \phi_1$. Assume that $\phi_1$ satisfies the property and

  $$\bigvee_{\sim_i^2((\pi_2,m),(\pi_2',m'))\neq 0} \big( \sim_i^2 ((\pi_2, m), (\pi_2', m')) \wedge [\phi]^{I_2}_{(\pi_2',m')} \big) = 0, \qquad (2.3)$$

for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Also, let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$, if such a path exists.

We have to prove that

$$\bigvee_{\sim_i^1((\pi_1,m),(\pi_1',m''))\neq 0} \sim_i^1 ((\pi_1, m), (\pi_1', m'')) \wedge [\phi]^{I_1}_{(\pi_1', m'')} = 0.$$

Let $(\pi_1', m'')$ be a point of $M_1$ such that $\sim_i^1 ((\pi_1, m), (\pi_1', m'')) \neq 0$. By 1, there exists $\pi_2' \in \mathsf{Paths}(M_2, \{\bot, 1\})$ a corresponding path to $\pi_1'$ and, by 2, we obtain $\sim_i^2 ((\pi_2, m), (\pi_2', m'')) \in \{\bot, 1\}$. Property (2.3) implies that $[\phi_1]^{I_2}_{(\pi_2', m'')} = 0$ and by the induction hypothesis we get $[\phi_1]^{I_1}_{(\pi_1', m'')} = 0$, which completes our proof. □

The following lemma can be proved in a similar way to Lemma 2.8.

**Lemma 2.9** Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ be an $(\alpha_R, \alpha_L, \alpha_S)$-abstraction by an equivalence $\rho$ of a multi-agent mv-Kripke structure $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ over $AP$ and $\mathcal{B}_3$, and $\phi$ an $\forall KCTL^*P_+$ formula. If

1. for any $\pi_2 \in \mathsf{Paths}(M_2, \{\bot, 1\})$ there exists a path $\pi_1 \in \mathsf{Paths}(M_1, \{\bot, 1\})$ with $\pi_1 \in C_{M_1}(\pi_2)$;
2. for any $q, q' \in Q_1$, $\sim_i^2 ([q], [q']) \in \{\bot, 1\}$ implies $\sim_i^1 (q, q') \in \{\bot, 1\}$;
3. for any $q, q' \in Q_1$, $R_2([q], [q']) \in \{\bot, 1\}$ implies $R_1(q, q') \in \{\bot, 1\}$;
4. for any $q \in Q_1$ and any $p \in AP$, $L_2([q])(p) = 0$ implies $L_1(q)(p) = 0$;

then

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$ ($I_1$, $I_2$ are the interpreted systems corresponding to $M_1$ and $M_2$, respectively).

**Proof** Again, the proof can be done by structural induction on $\phi$. We will detail only the cases that do not appear in the previous lemma.

- $\phi = \forall \phi_1$. Assume that $\phi_1$ satisfies the property and

$$[\forall \phi_1]^{I_2}_{(\pi_2,m)} = \bigwedge_{\pi[1..m]=\pi_2[1..m]} [\phi_1]^{I_2}_{(\pi,m)} = 0,$$

for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$. Then, there exists a path $\pi_2'$ with $\pi_2'[1..m] = \pi_2[1..m]$ such that $[\phi_1]^{I_2}_{(\pi_2', m)} = 0$. Let $\pi_1$ be a path of $M_1$ such that $\pi_1 \in C_{M_1}(\pi_2)$. Applying the induction hypothesis, we get that $[\phi_1]^{I_1}_{(\pi_1', m)} = 0$, for any path $\pi_1' \in C_{M_1}(\pi_2')$. Because of the third condition and $\pi_2'[1..m] = \pi_2[1..m]$, there exists $\pi_1'' \in C_{M_1}(\pi_2')$ such that $\pi_1''[1..m] = \pi_1[1..m]$ and $[\phi_1]^{I_1}_{(\pi_1'', m)} = 0$ which implies $[\forall \phi_1]^{I_1}_{(\pi_1, m)} = 0$;

- $\phi = \mathbf{K}_i \phi_1$. Assume that $\phi_1$ satisfies the property and

$$\bigwedge_{\sim_i^2((\pi_2,m),(\pi_2',m'))\neq 0} \sim_i^2 ((\pi_2, m), (\pi_2', m')) \wedge [\phi]^{I_2}_{(\pi_2', m')} = 0,$$

for some point $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

Then, there exists some point $(\pi_2', m') \in \mathsf{Points}(M_2, \{\bot, 1\})$ such that $\sim_i^2 ((\pi_2, m), (\pi_2', m')) \in \{\bot, 1\}$ and we have $[\phi]^{I_2}_{(\pi_2', m')} = 0$. Let $\pi_1 \in C_{M_1}(\pi_2)$ and $\pi_1' \in C_{M_1}(\pi_2')$ (by 1, we know that they exist). By 2, we have that $\sim_i^1 ((\pi_1, m), (\pi_1', m')) \in \{\bot, 1\}$ and by the induction hypothesis, we obtain that $[\phi]^{I_1}_{(\pi_1', m')} = 0$. Consequently, $[\mathbf{K}_i \phi_1]^{I_1}_{(\pi_1, m)} = 0$. □

In the following, we will state the error preservation results.

**Theorem 2.28** Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure over $AP$ and $\mathcal{B}_3$, $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ an $(\alpha_3, \alpha_L, \alpha_3)$-abstraction by an equivalence $\rho$ with $\alpha_L \in \{\alpha_1, \alpha_3\}$, and $\phi$ an $\exists KCTL^*P_+$ formula. Then

$$([\phi]^{I_2}_{(\pi_2, m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1, m)} = 0),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemma 2.8. □

Next, we prove that the $(\alpha_2, \alpha, \alpha_2)$-abstractions, for some $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$, are error preserving with respect to $\forall KCTL^*P$. Again, $\rho$ must be compatible with $\sim_i^1$.

**Theorem 2.29** Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure over $AP$ and $\mathcal{B}_3$, $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ an $(\alpha_2, \alpha_L, \alpha_2)$-abstraction by an equivalence $\rho$ compatible with $\sim_i^1$ with $\alpha_L \in \{\alpha_1, \alpha_3\}$, and $\phi$ an $\forall KCTL^*P_+$ formula. Then

$$([\phi]^{I_2}_{(\pi_2, m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1, m)} = 0),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemma 2.9. □

**Theorem 2.30** Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \leq i \leq n))$ be a multi-agent mv-Kripke structure over $AP$ and $\mathcal{B}_3$, $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \leq i \leq n))$ an $(\alpha_1, \alpha_L, \alpha_1)$-abstraction by an equivalence $\rho$ with $\alpha_L \in \{\alpha_1, \alpha_3\}$, and $\phi$ an $\exists KCTL^*P_+$ formula. Then

$$([\phi]^{I_2}_{(\pi_2,m)} = 0) \Rightarrow (\forall \pi_1 \in C_{M_1}(\pi_2))([\phi]^{I_1}_{(\pi_1,m)} = 0),$$

for any $(\pi_2, m) \in \mathsf{Points}(M_2, \{\bot, 1\})$.

**Proof** Directly from Lemma 2.8. □

**Example 2.15** Using the predicate $odd(x)$, which is 1 for odd numbers and 0 otherwise, we transform $\phi$ into an equivalent formula from $\exists KCTL^*P_+$. The abstract system from Example 2.12 has only 4 reachable states and we can easily prove that the formula $\phi$ is false. Since $\phi \in \exists KCTL^*P_+$ we can apply the error preservation result from Theorem 2.28 and obtain that $\phi$ is false in the concrete system $I_1$.

# Chapter 3

# Abstractions of Data Types

Many abstraction techniques we have found in the literature are driven by almost the same mechanism (e.g., surjective functions) and based on similar property preservation results, but the formalisms used by authors are quite different. We may say that all these abstraction techniques have their roots in Cousot's abstract interpretation framework [33], but this framework offers only a general methodology which, in particular cases should be complemented by specific techniques. Therefore, in our opinion, the development of specialized abstraction formalisms to allow reasonable instantiations in practical cases, is necessary.

In this chapter, we provide a solution to this problem with respect to data types, which extends the one in [111]. It tries to capture the essence of data type reduction and, in order to do that (abstract) data types are modeled by membership algebras enriched by sets of predicate symbols. This is a widely accepted formalism for specifying data types which offers mathematical precision and, on the other side, is practical related in that that many modern programming languages, such as C++ and Java, allow the users to define abstract data types beyond the basic ones. We define an abstraction as being a pair consisting of a congruence and an interpretation policy. The congruence partitions the original data type and redefines its operations in order to operate properly on the quotient data type. The interpretation policy interprets the predicate symbols into the quotient data type. It is shown that the interpretation policy cannot be substituted by the congruence as in the case of the operations. Therefore, a data type abstraction should necessarily include an interpretation policy.

The definition of an abstraction we adopted has proved to be very suitable from many points of view. First, we were able to classify abstractions based on the property preservation they assure. This classification shows clearly that the property preservation an abstraction assures depends directly on

the interpretation policy and indirectly on the congruence. Secondly, the abstraction technique proposed in the paper generalizes and clarifies the nature of many abstraction techniques found in the literature, such as the technique of duplicating predicate symbols [26, 35, 6], shape analysis [96, 104], predicate abstraction [57, 36, 113], McMillan's approach [86] etc. For example, it is shown that the technique of duplicating predicate symbols, which is based on associating two versions to each formula, one used for validation and the other one used for refutation, consists of two abstractions based on the same congruence. One is used in conjunction with validation formulas (because these abstractions preserve the truth value 1), and one is used in conjunction with refutation formulas (because these abstractions preserve the truth value 0). Therefore, the nature of this technique is clearly emphasized.

The abstraction technique proposed in the paper scale well from data types to abstract data types. Here, abstractions are applied to initial specifications by means of equations. The result of such an abstraction is a specification of a quotient abstract data type, which is in fact a new abstract data type. Therefore, analysis techniques specific to abstract data types can be combined with the abstraction technique proposed in this paper and applied in order to reason on (quotient) abstract data types.

The chapter is organized as follows. The first section recalls basic concepts on membership algebras and the second one provides a very brief introduction to (abstract) data types, motivates the usefulness of the membership algebra apparatus in their study, and fixes the logic to be used in order to reason about data types. Abstractions of data types are introduced in the third section, where property preservation results are also provided. The comparisons with other abstraction techniques are the subject of the next section. Thus, it is shown that the technique of duplicating predicate symbols, shape analysis, predicate abstraction and McMillan's approach are all particular cases of our approach (from the abstraction's point of view). The last section extends the abstraction technique discussed in the previous sections to abstract data types.

## 3.1 Preliminaries on Membership Algebra

We start the presentation of the abstraction techniques for abstract data types by recalling a few concepts regarding membership algebra [91, 8].

Let $K$ be a non-empty set whose elements will be called *kinds*. In this framework, the word kind is used instead of the more usual word sort, that will be reserved for another purpose. A *$K$-kinded membership signature* is a pair $\Omega = (\Sigma, \pi)$ which consists of a $(K^* \times K)$-indexed family of function

symbols $\Sigma = (\Sigma_{w,k} | (w,k) \in K^* \times K)$ such that $\Sigma_{w,k} \cap \Sigma_{w,k'} = \emptyset$ for any $(w,k),(w,k') \in K^* \times K$ with $k \neq k'$, and a function $\pi : S \to K$, where $S$ is a non-empty set disjoint of $K$ whose elements are called *sorts* . The elements $(w,s) \in K^* \times K$ are called *types* over $K$, and the elements $\sigma \in \Sigma_{w,k}$ are called *function or operation symbols of type* $(w,k)$; the elements $\sigma \in \Sigma_{\lambda,k}$ are also called *constant symbols of kind* $k \in K$. $\Sigma_{\lambda,k}$ is re-denoted by $\Sigma_k$, for any $k \in K$.

Let $\Omega = (\Sigma, \pi)$ be a $K$-kinded membership signature. Regarding kinds as sorts, $\Sigma$ can be thought as an ordinary signature. In this context, any algebra of signature $\Sigma$ will also be called a *$K$-kinded $\Sigma$-algebra*. That is, a $K$-kinded $\Sigma$-algebra is a pair $\mathbf{A} = (A, \Sigma^A)$, where $\Sigma^A = (\Sigma^A_{w,k} | (w,k) \in K^* \times K)$, $\Sigma^A_{w,k} = \{\sigma^A | \sigma \in \Sigma_{w,k}\}$, and $\sigma^A$ is a function from $A_w$ into $A_k$, for all $(w,k) \in K^* \times K$ and $\sigma \in \Sigma_{w,k}$ ($A_w$ denotes $\{\emptyset\}$, if $w = \lambda$, and $A_{k_1} \times \cdots \times A_{k_n}$, if $w = k_1 \cdots k_n \in K^+$). An *$\Omega$-algebra* is a triple $\mathcal{A} = (A, \Sigma^A, \Pi_A)$, where $(A, \Sigma^A)$ is $K$-kinded $\Sigma$-algebra and $\Pi_A$ is a function which assigns to each sort $s \in S$ a subset $A_s \subseteq A_{\pi(s)}$. We will denote $\mathbf{A} = (A, \Sigma^A)$ and call it the *$K$-kinded $\Sigma$-algebra associated to $\mathcal{A}$*.

An *$\Omega$-homomorphism* from an $\Omega$-algebra $\mathcal{A}$ to an $\Omega$-algebra $\mathcal{B}$ is a $\Sigma$-homomorphism $h$ from $\mathbf{A}$ to $\mathbf{B}$ such that $h_{\pi(s)}(A_s) \subseteq B_s$, for any sort $s \in S$. *$\Omega$-equivalences* (*$\Omega$-congruences*) on $\mathcal{A}$ are $\Sigma$-equivalences ($\Sigma$-congruences) on $\mathbf{A}$.

Given an $\Omega$-algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ and an $\Omega$-congruence $\rho = (\rho_k | k \in K)$, the *quotient of $\mathcal{A}$ by $\rho$* is the $\Omega$-algebra $\mathcal{A}/\rho = (A/\rho, \Sigma^{A/\rho}, \Pi_{A/\rho})$, where $\mathbf{A}/\rho = (A/\rho, \Sigma^{A/\rho})$ is the quotient of the $\Sigma$-algebra $\mathbf{A}$ by $\rho$ and $\Pi_{A/\rho}(s) = \{[a]_{\rho_{\pi(s)}} | [a]_{\rho_{\pi(s)}} \cap \Pi_A(s) \neq \emptyset\}$, for any sort $s$.

Given a $K$-kinded signature $\Sigma$ and a disjoint family $X = (X_k | k \in K)$ of variables, disjoint of $\Sigma$ as well, denote by $T_\Sigma(X)_k$ the set of *terms of kind $k$ over $\Sigma$ and $X$*. The $K$-indexed set $T_\Sigma(X) = (T_\Sigma(X)_k | k \in K)$ can be structured as a $K$-kinded $\Sigma$-algebra as usual, denoted $\mathbf{T}_\Sigma(X)$. When $X = \emptyset$, we simply write $\mathbf{T}_\Sigma$ instead of $\mathbf{T}_\Sigma(\emptyset)$. The elements of this algebra are called *ground terms*. $\mathbf{T}_\Sigma$ is an *initial algebra* in the class $Alg_\Sigma$ of all $\Sigma$-algebras, that is, there exists a unique $\Sigma$-homomorphism $eval_{\mathbf{A}}$ from $\mathbf{T}_\Sigma$ into $\mathbf{A}$, for any $\mathbf{A} \in Alg_\Sigma$.

If $\Omega = (\Sigma, \pi)$ is a $K$-kinded membership signature and $\Pi$ is a function which assigns a subset of $T_\Sigma(X)_{\pi(s)}$ to any sort $s \in S$, then $\mathcal{T}_\Omega(X, \Pi) = (T_\Sigma(X), \Sigma^{T_\Sigma(X)}, \Pi)$ is an $\Omega$-algebra. $\mathcal{T}_\Omega(\Pi)$ stands for $\mathcal{T}_\Omega(\emptyset, \Pi)$.

An *assignment* of $X$ into an $\Omega$-algebra $\mathcal{A}$ is a $K$-indexed set of functions $\gamma = (\gamma_k | k \in K)$ such that $\gamma_k$ is a function from $X_k$ into $A_k$, for all $k \in K$. If $a \in A_k$ and $x \in X_k$, then $\gamma[x/a]$ is the assignment obtained from $\gamma$ by replacing the value $\gamma_k(x)$ by $a$. $\bar{\gamma}$ denotes the unique homomorphic extension of $\gamma$ to $\mathbf{T}_\Sigma(X)$, and $\Gamma(X, \mathcal{A})$ stands for the set of all assignments of $X$ into

$\mathcal{A}$ ($\bar{\gamma}$ may not be a homomorphism from $\mathcal{T}_\Omega(X,\Pi)$ to $\mathcal{A}$, for any $\Pi$ as above).

In the *membership equational logic* over $\Omega = (\Sigma, \pi)$ and $X$ there are two types of *atomic formulas*:

- *equations*, which are of the form $t = t'$, where $t, t' \in T_\Sigma(X)_k$ for some kind $k$, and
- *membership assertions*, which are of the form $t : s$, where $s \in S$ and $t \in T_\Sigma(X)_{\pi(s)}$.

A *sentence* (*formula*) in the membership equational logic is a universally quantified Horn clause on the above atomic formulas, that is, a sentence of the form "$e$ `if` $\mathcal{C}$", where $e$ is an atomic formula and $\mathcal{C}$ is a finite set of atomic formulas.

Given an $\Omega$-algebra $\mathcal{A}$, the satisfaction of atomic formulas with respect to a given assignment $\gamma : X \to A$ is defined by:

- $\mathcal{A}, \gamma \models t = t'$ iff $\bar{\gamma}_k(t) = \bar{\gamma}_k(t')$;
- $\mathcal{A}, \gamma \models t : s$ iff $\bar{\gamma}_{\pi(s)}(t) \in A_s$.

Given a sentence $\phi = e$ `if` $\mathcal{C}$, we say that $\mathcal{A}$ *satisfies* $\phi$ or $\mathcal{A}$ is a *model of* $\phi$, denoted $\mathcal{A} \models \phi$, if $\gamma$ satisfies $e$ whenever it satisfies each atomic formula in $\mathcal{C}$, for any $K$-kinded assignment $\gamma : X \to A$. For a set $\Phi$ of sentences we write $\mathcal{A} \models \Phi$ iff $\mathcal{A} \models \phi$, for all $\phi \in \Phi$. The class of all $\Omega$-algebras that are models for a set $\Phi$ of sentences is denoted by $Mod(\Phi)$.

Given a set $\Phi$ of sentences over $\Omega = (\Sigma, \pi)$ and $X$, the binary relation $=_\Phi$ on $T_\Sigma(X)$ given by $t =_{\Phi,k} t'$ iff $(\forall \mathcal{A} \in Mod(\Phi))(\forall \gamma : X \to A)(\bar{\gamma}_k(t) = \bar{\gamma}_k(t'))$, for any kind $k$ and terms $t$ and $t'$ of kind $k$, is a congruence. Moreover, if we consider the function $\Pi : S \to 2^{T_\Sigma(X)}$ given by

$$\Pi(s) = \{[t] \in T_\Sigma(X)_{\pi(s)}/=_{\Phi,\pi(s)} \mid (\forall \mathcal{A} \in Mod(\Phi))(\forall \gamma : X \to A)(\bar{\gamma}_{\pi(s)}(t) \in A_s)\},$$

for any $s \in S$, then $(T_\Sigma(X)/=_\Phi, \Sigma^{T_\Sigma(X)/=_\Phi}, \Pi)$ is an $\Omega$-algebra. As this algebra is uniquely defined by $\Phi$, we denote it by $\mathcal{T}_{\Omega,\Phi}(X)$.

According to [91], $\mathcal{T}_{\Omega,\Phi}(X)$ is a *free algebra* in the class $Mod(\Phi)$, i.e., for any $\Omega$-algebra $\mathcal{A} \in Mod(\Phi)$ and for each assignment $\gamma : X \to A$ there exists a unique $\Omega$-homomorphism $h : \mathcal{T}_{\Omega,\Phi}(X) \to \mathcal{A}$ such that $h_k([t]_{=_{\Phi,k}}) = \bar{\gamma}_k(t)$, for any kind $k$ and any term $t$ of kind $k$. Moreover, $\mathcal{T}_{\Omega,\Phi} = \mathcal{T}_{\Omega,\Phi}(\emptyset)$ is an *initial algebra* in the class $Mod(\Phi)$, i.e., for any $\Omega$-algebra $\mathcal{A} \in Mod(\Phi)$ there exists a unique $\Omega$-homomorphism $h : \mathcal{T}_{\Omega,\Phi} \to \mathcal{A}$.

The following set of deduction rules is sound and complete for the membership logic over $\Omega$ and $X$ [91]:

**subject reduction:** $\dfrac{t : s,\ t = t'}{t' : s}$

**membership:** $\dfrac{t_1 : s_1, \ldots, t_n : s_n, u_1 = v_1, \ldots, u_n = v_n}{t : s}$, where $t : s$ `if` $t_1 : s_1, \ldots, t_n : s_n, u_1 = v_1, \ldots, u_n = v_n \in \Phi$

**reflexivity:** $\dfrac{}{t = t}$

**symmetry:** $\dfrac{t = t'}{t' = t}$

**transitivity:** $\dfrac{t = t',\ t' = t''}{t = t''}$

**congruence:** $\dfrac{t_1 = t'_1, \ldots, t_n = t'_n}{f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)}$, where $t_i$ and $t'_i$ are terms of kinds $k_i$, $1 \leq i \leq n$, and $f$ is of type $(k_1, \ldots k_n, k)$

**replacement:** $\dfrac{t_1 : s_1, \ldots,\ t_n : s_n,\ u_1 = v_1, \ldots,\ u_n = v_n}{t = t'}$, where $t = t'$ `if` $t_1 : s_1, \ldots, t_n : s_n, u_1 = v_1, \ldots, u_n = v_n \in \Phi$

## 3.2 Reasoning About Data Types

We model data types by membership algebras, a widely accepted formalism for specifying data types [91, 8]. To address the problem of verifying or analyzing a particular program that uses a certain data structure, we enrich signatures with logical symbols used to build formulas. Questions about properties of data structures will be answered by evaluating such formulas.

**(Abstract) Data Types** In this section we provide a brief introduction to (abstract) data types and motivate the usefulness of the membership algebra apparatus, an extension of universal algebra, in their study. We follow mainly [40, 41, 95, 91, 8].

A *data type* consists of one or more sets of values, such as natural numbers, booleans, characters or strings, together with a collection of functions on these sets. Examples of basic data types provided by most programming languages include *integer*, *boolean*, *array*, *record*. From a mathematical point of view, a data type is an $\Omega$-algebra. The signature $\Sigma$ associates names to operations, while the algebra associates domains to kinds and interprets correspondingly the operation names. The sorts are used for overloading

operation names or for handling errors and partiality.

Many modern programming languages, such as C++ or Java, allow the user to define additional data types beyond the basic ones, such as *stack*, *queue*, *tree* or *counter*. As it is usual in such cases, operations are defined by *sentences*. For example, if we define a data type of stacks over arbitrary elements (stacks are sequences of elements and the empty stack will be denoted $\lambda$) with the operations $Push$ of type $(k_{stack}\, k_{elem}, k_{stack})$, $Pop$ of type $(k_{stack}, k_{stack})$, and $Top$ of type $(k_{stack}, k_{elem})$, and the sort *stack* of kind $k_{stack}$ to represent valid stacks, then the following sentences specify the properties of these operations:

1. $\lambda : stack$;
2. $Push(x, y) : stack$ `if` $x : element$ and $y : stack$;
3. $Top(Push(x, y)) = x$ `if` $y : stack$;
4. $Pop(Push(x, y)) = y$ `if` $y : stack$,

where $x$ is a variable of kind $k_{stack}$ and $y$ is a variable of kind $k_{elem}$. Notice that, by the specification of the sort *stack*, the term $Pop(\lambda)$ of kind $k_{stack}$ is not of sort *stack* and it will be considered an *error*. Such data types are usually called *abstract data types*. In fact, a *specification* as the one above stands for all data types of stacks over equipotent [1] sets of elements. These data types are "similar" in that they differ only by the "nature" of their elements. From a mathematical point of view, an abstract data type is a class of $\Omega$-algebras closed under isomorphism (the isomorphism models the "similarity" concept above). More about abstract data types and specifications is provided in Section 3.5.

There are many advantages which follow from the utilization of the membership algebra apparatus in modeling (abstract) data types, such as:

- mathematical precision;
- using membership algebras, instead of universal algebras, we can deal well with errors and partiality;
- independence of any particular implementation in a computer language. Therefore, we may reason about effects of the operations, relations to other data types etc.;

[1] Two sets are equipotent if there exists a bijective function from one set into the other one.

- easiness in defining new operations and predicates on data types. For example, from the axioms above, one may easily define the predicate $IsEmpty(x)$ by the following additional axioms:
  - $IsEmpty(\lambda) = true$;
  - $IsEmpty(Push(x, y)) = false$;
- easiness in checking program correctness. If a program using a set of specified data types is designed so that the correctness of the program depends only on the specification, then the primary concern of the data type implementor is to satisfy the specification. In this way, neither the user nor the implementor of a data type needs to worry about additional details of the other's program.
- membership algebra specifications can be efficiently implemented in systems like Maude [30].

Abstract data types are central to object-oriented programming where every "class" is an abstract data type. An "object" is a data structure encapsulated with a set of routines, called "methods", which operate on the data. Operations on the data can only be performed via these methods, which are common to all objects that are instances of a particular class. Thus the interface to objects is well defined, and allows the code implementing the methods to be changed as long as the interface remains the same.

**A Motivating Example** We consider a toy example in order to motivate the concepts we are going to introduce. More developed examples are provided later in the chapter.

**Example 3.1** The set of natural numbers together with the addition operation defines a data type that can be modeled by a $K$-kinded $\Omega$-algebra $\mathcal{A}$, where:

- $K$ contains only one kind (denoted by $nat$) and $S$ is empty;
- $\Sigma$ contains one constant symbol of type $nat$ for each natural number, and one operation symbol $+$ of type $(nat\, nat, nat)$;
- $A_{nat} = \mathbf{N}$;
- the constant symbol associated to a natural number is interpreted as the number itself, and $+^A$ is the usual addition operation on natural numbers.

In this algebra, the following property trivially holds:

$$(\varphi_1)\ \ (\forall x, y \in A_{nat})(Isgrz^A(x)\ \vee\ Isgrz^A(y)\ \Rightarrow\ Isgrz^A(x +^A y)),$$

where $Isgrz$ is a unary predicate symbol whose meaning is "is greater than zero", $Isgrz^A$ is its interpretation in $\mathcal{A}$, '$\vee$' and '$\Rightarrow$' are the usual "or" and "implies" predicates, $x$ and $y$ are variables, and '$\forall$' is the universal quantifier. These new elements do not belong to the algebra $\mathcal{A}$; they are part of a *meta-language* used to express properties of $\mathcal{A}$.

Assume now that we are interested in distinguishing between 0 and the other natural numbers. That is, we want to treat all the natural numbers greater than 0 as a whole. In order to do that we define a congruence $\rho$ by

$$a\,\rho\,b \ \text{ iff } \ \text{either } a = b = 0 \text{ or } a, b \neq 0,$$

for all natural numbers $a$ and $b$. The equivalence classes induced by $\rho$ are $[0]$, containing only the number 0, and $[1]$, containing all the other numbers. The addition operation on these equivalence classes is given in the table below.

| $+^{A/\rho}$ | $[0]$ | $[1]$ |
|---|---|---|
| $[0]$ | $[0]$ | $[1]$ |
| $[1]$ | $[1]$ | $[1]$ |

The algebra $\mathcal{A}/\rho$ acts as an abstraction of $\mathcal{A}$ with respect to the property mentioned above (all the natural numbers greater than 0 are treated as a whole). The predicate symbol $Isgrz$ can be interpreted in $A/\rho$ by

$$Isgrz^{A/\rho}([a]) \quad \text{iff} \quad (\forall a' \in [a])(a' > 0),$$

for any $a$. Now, the property $\varphi_1$ can be rewritten as follows:

$$(\varphi_2) \quad (\forall x, y \in A_{nat}/\rho)(Isgrz^{A/\rho}(x)\ \vee\ Isgrz^{A/\rho}(y)\ \Rightarrow$$
$$Isgrz^{A/\rho}(x +^{A/\rho} y)).$$

We have to remark that the validity of $\varphi_2$ in $\mathcal{A}/\rho$ leads to the validity of $\varphi_1$ in $\mathcal{A}$. Moreover, $\varphi_2$ holds in $\mathcal{A}/\rho$, and this can be easily checked out by an automatic procedure due to the fact that $\mathcal{A}/\rho$ contains only two elements. Therefore, $\varphi_1$ holds in $\mathcal{A}$. Much more, $\varphi_1$ and $\varphi_2$ are in fact interpretations of the same formula $\varphi$ (of the meta-language)

$$(\varphi) \quad (\forall x, y)(Isgrz(x)\ \vee\ Isgrz(y)\ \Rightarrow\ Isgrz(x + y))$$

in two different algebras of the same signature. We can say that the validity of $\varphi$ in $\mathcal{A}/\rho$ implies the validity of $\varphi$ in $\mathcal{A}$.

Our example works fine with the predicates we considered. However, for other predicates things can be totally different. Let us consider, for instance, the equality predicate on $A$. It can be interpreted in $\mathcal{A}/\rho$ in various ways. One of the most natural way is to consider a new truth value $\perp$ whose meaning is "indefinite", and to define the predicate as in the table below.

| $=^{A/\rho}$ | [0] | [1] |
|---|---|---|
| [0] | 1 | 0 |
| [1] | 0 | $\perp$ |

$(=^{A/\rho}([1],[1])$ is evaluated to $\perp$ because two arbitrary numbers in [1] can be equal or different). $\square$

**Logically Extended Signatures** The example in the paragraph above leads us to the following considerations:

1. the meta-language used to express properties of data types (algebras) should be specific to signatures and not to data types (algebras). But, even if the elements of the meta-language are specific to signatures they should work properly inside the algebras.

2. data type reductions can be captured by congruences. In such a case, the operations are automatically redefined to operate on the quotient data type (algebra), but the predicates need a special treatment (more arguments about this are provided in Section 3.3).

We will discuss (1) in this section, and (2) in the next section.

Let $K$ be a set of kinds. *Logically extended $K$-kinded membership signatures* are adding *logical symbols* (*predicate symbols*) to ordinary membership signatures. The logical symbols of such a signature have two roles:

– to specify basic properties satisfied by elements of an algebra;

– to build formulas defining new properties.

**Definition 3.1** A *logically extended $K$-kinded membership signature* is a 4-tuple $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$, where:

- $\mathcal{B} = (B, \wedge, \vee, \neg)$ is a truth algebra;
- $\leq'$ is a partial order on $B$ such that:
  - $(B, \leq')$ is an *inf-complete lattice*, i.e. any subset $B' \subseteq B$ has a greatest lower bound. We denote by $0'$ its smallest element;

– for any $x \in B$ there exists a unique $y \in B$ such that $x \succ' y$ ($<'=\leq' - \{(a,a)|a \in B\}$, $>'$ is the inverse of $<'$, and $x \succ' y$ if $x >' y$ and there exists no $c \in B$ such that $x >' c >' y$).

- $K' \subseteq K$ is a set of *basic kinds*.
- $\Sigma_L = (\Sigma_{L,w}|w \in K'^+)$ is a set of *predicate symbols* ($w$ is called a *logical type*);
- the set of kinds $K$ contains apart from $K'$, a kind for each logical type for which we have at least one predicate symbol in $\Sigma_L$. Formally,

$$K = K' \cup \{k_w|w \in K'^+ \text{ and } \Sigma_{L,w} \neq \emptyset\};$$

- $\Sigma$ is a $K^* \times K$-indexed family of function symbols such that it contains a function symbol $(\_ , \cdots , \_)$ for each logical type for which we have at least one predicate symbol in $\Sigma_L$. The function symbol $(\_ , \cdots , \_)$ associated to the logical type $w = k_1 \cdots k_m \in K'^+$ has the arity $(k_1 \cdots k_m, k_w)$;
- the set of sorts $S$ contains a distinguished sort $s_{p,b}$, for each $p \in \Sigma_L$ and $b \in B - \{0'\}$.
- $\pi$ is a function from $S$ into $K$ such that $\pi(s_{p,b}) = k_w$, for any $p \in \Sigma_{L,w}$, $b \in B - \{0'\}$ and $w \in K'^+$.

The kind $k_w$, for some logical type $w = k_1 \ldots k_m \in K'^+$, represents all possible $m$-tuples for which the $i$th element is of kind $k_i$ (by means of the corresponding function symbol $(\_ , \cdots , \_)$) and $s_{p,b}$ defines the set of all tuples on which $p$ gets truth values greater than or equal to $b$ (with respect to $\leq'$), for any $b \in B - \{0'\}$. The tuples on which $p$ is $0'$ are the ones that do not belong to any $s_{p,b}$ with $b \in B - \{0'\}$. In fact, the representation we have used for multi-valued predicates is obtained in two steps:

- we represent a multi-valued predicate $p$ by a set of two-valued predicates $\{p_b|b \in B - \{0'\}\}$ such that $p_b$ is 1 for elements on which $p$ is greater than or equal to $b$ (with respect to $\leq'$);
- each 2-valued predicate is represented by a sort that specifies it's non-zero values.

A logically extended membership signature should be thought as an ordinary membership signature that contains some distinguished kinds and

some distinguished sorts; the predicate symbols are just used to specify the distinguished kinds and sorts.

Logically extended algebras are membership algebras that correspond to logically extended membership signatures such that the meaning of the sorts representing predicates is as above.

**Definition 3.2** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a logically extended $K$-kinded signature. An $\Omega_L$-*algebra* is a membership algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ such that:

- $\Pi_A(s_{p,b}) \subseteq \Pi_A(s_{p,b'})$, for any $p \in \Sigma_L$ and $b, b' \in B - \{0'\}$ with $b' \leq' b$.
- $\Pi_A(s_{p,b}) \cap \Pi_A(s_{p,b'}) = \emptyset$, for any $p \in \Sigma_L$ and $b, b' \in B - \{0'\}$ such that there exists $x \in B$ with $x \prec' b$ and $x \prec' b'$.

By the representation of multi-valued predicates used in the logically extended membership algebras, the truth value of some predicate $p \in \Sigma_{L,w}$ over an element $a \in A_{k_w}$ is the maximum truth value $b \in B - \{0'\}$, with respect to $\leq'$, for which $a \in \Pi_A(s_{p,b})$, or $0'$ if such a $b$ does not exist.

Homomorphisms, equivalences, and congruences of $\Omega_L$-algebras are homomorphisms, equivalences, and congruences, respectively, of membership algebras.

Given $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ a logically extended $K$-kinded membership signature and a (denumerable) set $X$ of $K$-kinded variables, we define the set of *first order formulas* over $\Omega_L$ and $X$ as in Section 1.2.1 but, now, the elements of the universe are terms:

1. *atomic formulas:*
   (a) $p(t_1, \cdots, t_m)$ is an atomic formula, for any logical symbol $p$ of type $k_1 \cdots k_m$ and any term $t_i$ of kind $k_i$, $1 \leq i \leq m$;
2. *formulas:*
   (a) every atomic formula is a formula;
   (b) if $\alpha$ and $\beta$ are formulas, then $(\alpha \vee \beta)$, $\neg\alpha$, and $(\alpha \wedge \beta)$ are formulas;
   (c) if $x$ is a variable and $\alpha$ is a formula, then $((\exists x)\alpha)$ and $((\forall x)\alpha)$ are formulas.

Denote by $\mathcal{L}^O(\Omega_L, X)$, where $O \subseteq \{\wedge, \vee, \neg\}$, the set of first order formulas over $\Omega_L$ and $X$ that use only the operators from the set $O$, the quantifier $\forall$ if $\wedge \in O$ and the quantifier $\exists$ if $\vee \in O$.

Given an $\Omega_L$-algebra $\mathcal{A}$, each term or first order formula $\varphi$ induces a function $\mathcal{I}_{\mathcal{A}}(\varphi)$ from the set $\Gamma(X, \mathcal{A})$ (of all assignments of $X$ into $\mathcal{A}$, which are defined as for membership algebras) into $A \cup B$, as follows:

- $\mathcal{I}_{\mathcal{A}}(x)(\gamma) = \gamma(x)$, for any variable $x$;
- $\mathcal{I}_{\mathcal{A}}(\sigma)(\gamma) = \sigma^A$, for any constant symbol $\sigma$;
- $\mathcal{I}_{\mathcal{A}}(\sigma(t_1, \ldots, t_m))(\gamma) = \sigma^A(\mathcal{I}_{\mathcal{A}}(t_1)(\gamma), \cdots, \mathcal{I}_{\mathcal{A}}(t_m)(\gamma))$, for any term $\sigma(t_1, \cdots, t_m)$;
- $\mathcal{I}_{\mathcal{A}}(p(t_1, \ldots, t_m))(\gamma)$, for any atomic formula $p(t_1, \ldots, t_m)$, is the truth value of $p$ over $(\mathcal{I}_{\mathcal{A}}(t_1)(\gamma), \cdots, \mathcal{I}_{\mathcal{A}}(t_m)(\gamma))$;
- $\mathcal{I}_{\mathcal{A}}(\psi_1 \wedge \psi_2) = \mathcal{I}_{\mathcal{A}}(\psi_1) \wedge \mathcal{I}_{\mathcal{A}}(\psi_2)$ and $\mathcal{I}_{\mathcal{A}}(\psi_1 \vee \psi_2) = \mathcal{I}_{\mathcal{A}}(\psi_1) \vee \mathcal{I}_{\mathcal{A}}(\psi_2)$, for any formulas $\psi_1, \psi_2$;
- $\mathcal{I}_{\mathcal{A}}(\neg\psi) = \neg\mathcal{I}_{\mathcal{A}}(\psi)$, for any formula $\psi$;
- $\mathcal{I}_{\mathcal{A}}((\exists x)\psi)(\gamma) = \bigvee_{a \in A_k} \mathcal{I}_{\mathcal{A}}(\psi)(\gamma[x/a])$ and $\mathcal{I}_{\mathcal{A}}((\forall x)\psi)(\gamma) = \bigwedge_{a \in A_k} \mathcal{I}_{\mathcal{A}}(\psi)(\gamma[x/a])$, for any formula $\psi$ and any variable $x$ of kind $k$.

$\mathcal{I}_{\mathcal{A}}(\psi)$ is called the *interpretation function* of $\psi$ into $\mathcal{A}$. If $\psi$ is a formula, we say that $\psi$ has the truth value $b \in B$ in $\mathcal{A}$, and denote this by $[\psi]_{\mathcal{A}} = b$, if $\mathcal{I}_{\mathcal{A}}(\psi)(\gamma) = b$, for all $\gamma \in \Gamma(X, \mathcal{A})$.

**Example 3.2** The algebra in Example 3.1 can be simply transformed into an $\Omega_L$-algebra by considering $\Sigma_L = \{Isgrz\}$. Denote this algebra by $\mathcal{A}$ as well. The function $\gamma$ given by $\gamma(x) = 0$ and $\gamma(y) = 1$ is an assignment of $X$ into $\mathcal{A}$. Under this assignment, $\mathcal{I}_{\mathcal{A}}(\varphi)$ is valuated to 1, where $\varphi$ is the formula in Example 3.1. □

## 3.3 Abstractions of models

An algebra assigns a meaning to a signature by associating a set of data to each sort and an operation (function) to each operation symbol. Therefore, an algebra defines a concrete data type.

By a *data type reduction/abstraction* we understand to reduce types/domains of large or unbounded range to small types/domains (types/domains of small range). A few words about "small domains" are in order. First of all, we have to say that it is very hard to quantify the difference between large and small domains and, moreover, the difference is relative. In 1981,

when model checking was invented, systems with $10^{20}$ states were regarded as large systems. Nowadays, there are techniques that can be applied to systems with $10^{120}$ states. If for a class of systems verification techniques can be applied in reasonable time, the systems in the class can be regarded as small. Large systems are not necessarily infinite systems. Surely, infinite systems pose more problems than finite ones, although there are techniques for verifying particular infinite-state systems. As a conclusion, a system which allows practical verification can be regarded as a small one. Here, all the examples we are going to consider exhibit reductions to (finite) very small domains. However, we want to emphasize that our main goal is not to investigate abstractions leading to small domains. The main goal is twofold: first, we classify abstractions and show that many techniques proposed in the literature fall in one of the classes introduced in the paper, and secondly, we propose abstractions guided by equations.

As we have already mentioned in the previous section, an elegant way to formalize such reductions is to use congruences which are equivalence relations compatible with data types (algebras) operations. In this way, the operations of the original data type are automatically redefined to operate on the abstraction of the original data type.

In this section we study abstractions of data types; the extension to abstract data types will be studied in Section 3.5.

### 3.3.1 Abstractions

As mentioned in the previous section, congruences of logically extended algebras are defined as for membership algebras by taking into consideration the function symbols. The quotient is obtained as usual, all the sorts being defined by over-approximation.

In order to define abstractions we have to consider also interpretation policies and redefine the predicates according to them. Consequently, the abstractions induced by congruences are defined as usual quotients except for the sorts representing predicates.

**Definition 3.3** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$ and $\alpha$ an interpretation policy over $\mathcal{B}$. The *$\alpha$-abstraction of $\mathcal{A}$ by $\rho$* is an $\Omega_L$-algebra $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ such that:

- $\Pi'(s) = \{[a]_{\rho_{\pi(s)}} \mid [a]_{\rho_{\pi(s)}} \cap \Pi_A(s) \neq \emptyset\}$, for any sort $s$ not representing some predicate;

- $[a]_\rho \in \Pi'(s_{p,b})$, for some $p \in \Sigma_{L,w}$, $a \in A_{k_w}$ and $b \in B$, if there exists $b' \in B$ such that $b \leq' b'$ and $b'$ is the value of $p$ over $[a]_\rho$ according to $\alpha$.

**Remark 3.1** As we can see, the definition of an abstraction requires a special treatment of the sorts representing logical symbols. Including a carrier set for the range of the logical operations might appear as a good solution for a uniform treatment and this is the approach adopted in [92]. However, this may not allow to distinguish between predicate interpretations in the abstract system or may lead to useless abstractions, as it was already remarked in [111].

For example, let $\mathcal{A}$ be a membership algebra that models the set of natural numbers where the set $\{true, false\}$ is structured as a kind and a function symbol $\models$ is used with the meaning "$\models (e, p) = v$ iff $p$ has the truth value $v$ in the element $e$". Also, let $eq$ be the equality predicate defined on couples of natural numbers as usual:

$$\models ((n_1, n_2), eq) = true \text{ iff } n_1 = n_2.$$

We suppose that $\mathcal{A}$ contains a kind for couples of natural numbers and $(\_, \_)$ is a function symbol that constructs such couples.

Now, let $\rho$ be a congruence on $\mathcal{A}$ such that

$$a \,\rho\, b \text{ iff } a = b = 0 \text{ or } a, b \neq 0.$$

If we want to abstract $\mathcal{A}$ using $\rho$ we can obtain only *one* abstraction because $\models$ is a function symbol and it is automatically redefined in the quotient system by the congruence $\rho$. Moreover, we obtain:

$$\models^{\mathcal{A}/\rho} ([(1,1)], eq) = [\models^{\mathcal{A}} ((1,1), eq)] = [true] \text{ and}$$

$$\models^{\mathcal{A}/\rho} ([(1,2)], eq) = [\models^{\mathcal{A}} ((1,2), eq)] = [false]$$

which leads to $[true] = [false]$. That is, both truth values $true$ and $false$ are in the same equivalence class and the abstraction is useless.

Regarding the computation of multi-valued abstractions for logically extended membership algebras we can translate the results introduced for logical structures in Section 2.2.2. We can use the $S'$-congruences introduced in [46] to obtain some of the possible $\alpha$-abstractions of an $\Omega_L$-algebra. The $S'$-congruences are defined on membership algebras and allow sorts to be defined by under-approximation in the quotient algebra. If $A$ is a set, $\rho$ an equivalence on $A$, and $B$ a subset of $A$, we say that $B/\rho$ is defined by *over-approximation* if $B/\rho = \{[a] \in A/\rho | a \in B\}$, and we say that $B/\rho$ is defined by *under-approximation* if $B/\rho = \{[a] \in A/\rho | [a] \subseteq B\}$.

**Definition 3.4** Let $\Omega = (\Sigma, \pi)$ be a $K$-kinded membership signature and $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ be an $\Omega$-algebra. An *$S'$-congruence on $\mathcal{A}$* is a pair $(\rho, S')$ consisting of a congruence $\rho$ on $\mathcal{A}$ and a subset $S'$ of the set $S$ of sorts. *The quotient of $\mathcal{A}$ by $(\rho, S')$* is an $\Omega$-algebra $\mathcal{A}/(\rho, S') = (A/\rho, \Sigma^{A/\rho}, \Pi_{A/\rho})$, where $\Pi_{A/\rho}(s)$ is defined by under-approximation, if $s \in S'$, and by over-approximation, if $s \notin S'$.

$S'$-congruences can be applied as well on $\Omega_L$-algebras which are particular cases of membership algebras. Moreover, all the notions from above remain valid if we replace congruences by $S'$-congruences.

If $\mathcal{A}$ is an $\Omega_L$-algebra, we will make the connection between quotients of $\mathcal{A}$ relative to $S'$-congruences $(\rho, S')$ (which represent two-valued abstractions) and $\alpha$-abstractions of $\mathcal{A}$ by $\rho$ (which represent multi-valued abstractions), based on Theorem 2.8, in the following way:

- the set $S' \subseteq S$ has a similar property to the one of $\mathcal{P}$, that is, $s_{p,b} \in S'$ implies $(\forall p \in \Sigma_L)(s_{p,b} \in S')$. We will denote by $B_{S'}$ the set of truth values $b$ for which the sorts $s_{p,b}$ are in $S'$;
- we require that $\leq'$ and $S'$ are appropriate for the congruence $\rho$ (the appropriateness of $S'$ is defined similarly to the appropriateness of $\mathcal{P}$);
- we can prove, in a similar way to Theorem 2.7, that the quotient of an $\Omega_L$-algebra by an $S'$-congruence as above is also an $\Omega_L$-algebra.
- $\alpha$ is obtained using the eight rules of Theorem 2.8 in which we replace $B_{\mathcal{P}}$ with $B_{S'}$.

### 3.3.2 Property preservation

In this section we will prove the usefulness of the abstractions above by offering preservation results. These results will be translated from the multi-valued abstraction technique for logical structures presented in Section 2.2.

In fact, we will reduce the problem of checking the truth value of some formula $\phi$ in an $\Omega_L$-algebra $\mathcal{A}$ for some assignment $\gamma$ to the checking of the truth value of a formula $Tr(\phi)$ in a $(\mathcal{B}, \Sigma'_L)$-logical structure $\mathcal{S}_{\mathcal{A}}$ for some assignment $Tr(\gamma)$. Moreover, we will prove that this reduction preserves truth values, i.e.

$$\mathcal{I}_{\mathcal{A}}(\phi)(\gamma) = \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi))(Tr(\gamma)).$$

We start by describing the transformation that we apply on formulas. Let $\phi$ be a formula over $\Omega_L$ and some set of variables $X$. We say that a term $t$ is

*maximal* in $\phi$ if it is included in the following set:

$$\{t_1, \ldots, t_m | p(t_1, \ldots, t_m) \text{ a subformula of } \phi, \text{ for some } p\}.$$

To simplify the definition of the transformation $Tr(\phi)$ we suppose the following:

- any two atomic formulas are build using different predicate symbols;
- the quantified variables appear only in maximal terms that do not contain any other variable.

Clearly, if a formula does not satisfy the above we can add predicates and obtain an equivalent form with these properties.

The main idea of the transformation is the following: as in logical structures we are not interested in how elements from the universe are obtained, we will abstract the terms in $\phi$ by variables whose values will be the possible values of the corresponding terms. When we encounter quantified variables the transformation is a little bit more complicated: we will abstract all the terms in which some quantified variable appears by the same variable whose values will record possible values for all these terms. Obviously, to make these changes we also need new predicate symbols of appropriate logical types.

The formal definition of the transformation $Tr(\phi) \in \mathcal{L}(\Sigma'_L, X')$, for any formula $\phi \in \mathcal{L}(\Omega_L, X)$ with $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$, is as follows:

1. for any maximal term $t$ not containing a quantified variable, we add to $X'$ a new variable $x_t$ of a new kind $k_t$;
2. if $v$ is a quantified variable, suppose $v_1$,..., $v_n$ are the maximal terms that contain $v$. We will consider a new variable of a new kind $k_v$, denoted $(x_{v_1}, \ldots, x_{v_n})$;
3. if $p(t_1, \ldots, t_m)$ is an atomic formula without quantified variables then,

   $$Tr(p(t_1, \ldots, t_m)) = p'(x_{t_1}, \ldots, x_{t_m}),$$

   where $p'$ is a new predicate symbol of type $k_{t_1} \ldots k_{t_m}$.
4. if $p(t_1, \ldots, t_m)$ is an atomic formula such that the terms $t_{i_1}$,..., $t_{i_p}$, for some set of indexes $\{i_1, \ldots, i_p\} \subseteq \{1, \ldots, m\}$ contain the quantified variable $v$ then,

   $$Tr(p(t_1, \ldots, t_m)) = p_v(x_1, \ldots, x_m),$$

where

$$x_i = \begin{cases} (x_{v_1}, \ldots, x_{v_n}) & , \text{ if } i \in \{i_1, \ldots, i_p\}, \\ x_{t_i} & , \text{ otherwise.} \end{cases}$$

Above, $p_v$ is a new predicate symbol of type $k'_1 \ldots k'_m$, where

$$k'_i = \begin{cases} k_v & , \text{ if } i \in \{i_1, \ldots, i_p\}, \\ k_{t_i} & , \text{ otherwise.} \end{cases}$$

Because of the supposition we made on formulas, an extension to the case where more quantified variables appear in $p(t_1, \ldots, t_m)$ is straightforward.

5. $Tr$ leaves unchanged the logical operators $\neg$, $\wedge$ and $\vee$.

6. every quantifier $\forall v$ ($\exists v$) is replaced by $\forall(x_{v_1}, \ldots, x_{v_n})$ ($\exists(x_{v_1}, \ldots, x_{v_n})$), where $v_1$,..., $v_n$ are the maximal terms of $\phi$ that contain $v$.

**Remark 3.2** The transformation above does not add new logical operators, and, consequently, if $\phi \in \mathcal{L}^O(\Omega_L, X)$ then $Tr(\phi) \in \mathcal{L}^O(\Sigma'_L, X')$, for any $O \subseteq \{\wedge, \vee, \neg\}$.

If $K'$ is the set of kinds and $\Sigma'_L$ the set of predicates obtained by the transformation above then, to any $\Omega_L$-algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ we associate a $(\mathcal{B}, \Sigma'_L)$-logical structure $\mathcal{S}_\mathcal{A} = (S, \Sigma'^{\mathcal{S}_\mathcal{A}}_L)$ such that:

- $S_{k_t} = \{\mathcal{I}_\mathcal{A}(t)(\gamma) | \gamma \in \Gamma(X, \mathcal{A})\}$, for any term $t$. Notice that $S_{k_t}$ is a subset of $A_k$, where $k$ is the kind of $t$;

- $S_{k_v} = \{(\mathcal{I}_\mathcal{A}(v_1)(\gamma), \ldots, \mathcal{I}_\mathcal{A}(v_n)(\gamma)) | \gamma \in \Gamma(X, \mathcal{A})\}$, where $v_1$,..., $v_n$ are the maximal terms in $\phi$ that contain the quantified variable $v$;

- $p'^{\mathcal{S}_\mathcal{A}}(u_1, \ldots, u_m)$ is the value of $p$ over $(u_1, \ldots, u_m)$, for any predicate $p'$ added in the step 3 of the definition of $Tr(\phi)$.

- $p_v^{\mathcal{S}_\mathcal{A}}(u_1, \ldots, u_m)$ is the value of $p$ over $(u'_1, \ldots, u'_m)$, for any predicate $p_v$ added in the step 4 of the definition of $Tr(\phi)$, where $u'_1, \ldots, u'_m$ are obtained by the following considerations:

  - each $u_i$ with $i \in \{i_1, \ldots, i_p\}$ is a tuple $(a_1, \ldots, a_n)$, where $a_i = \mathcal{I}_\mathcal{A}(v_i)(\gamma)$, for some assignment $\gamma$. We take $u'_i = a_j$ if $t_i = v_j$;
  - $u'_i = u_i$ if $i \notin \{i_1, \ldots, i_p\}$.

Finally, each assignment $\gamma \in \Gamma(X, \mathcal{A})$ is transformed into an assignment $Tr(\gamma) \in \Gamma(X', \mathcal{S}_\mathcal{A})$ such that $Tr(\gamma)(x_t) = \mathcal{I}_\mathcal{A}(t)(\gamma)$ and $Tr(\gamma)((x_{v_1}, \ldots, x_{v_n})) = (\mathcal{I}_\mathcal{A}(v_1)(\gamma), \ldots, \mathcal{I}_\mathcal{A}(v_n)(\gamma))$.

**Example 3.3** Let $K$ be a set of kinds containing only one kind *nat* for the set of natural numbers and $\Omega_L = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L, \pi)$ a $K$-kinded logically extended signature with $\mathcal{B}_2$ the truth algebra containing only two values, $0 \leq' 1$, $\Sigma = \{0, Succ, +, \cdot, \%\}$ and $\Sigma_{L,nat\,nat} = \{p, q\}$ (0 is the constant representing the natural number 0, $Succ$ is the successor operation on natural numbers, $+$ is the addition, $\cdot$ is the multiplication and $\%$ is the modulo operation).

We take an $\Omega_L$-algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ such that $A_{nat} = \mathbf{N}$, $\Sigma^A$ interprets correspondingly the operation symbols from $\Sigma$, $\Pi_A(s_{p,1})$ is the set of couples $(a, b)$ with $a$ an even number and $\Pi_A(s_{q,1})$ is the set of couples $(a, b)$ with $a\%3 = 1$.

Moreover, consider the formula $\phi$ over $\Omega_L$ and $X = \{x, y, z\}$ as follows:

$$\phi = \forall x \exists y \big( (p(x\%2, y) \vee p(y + 3, z\%5)) \wedge q(4 \cdot z + 5, 9) \big).$$

Above, we have used 2 instead of the term $Succ(Succ(0))$, 3 instead of $Succ(Succ(Succ(0)))$, etc.

If we denote $t_1 = x\%2$, $t_2 = y$, $t_3 = y + 3$, $t_4 = z\%5$, $t_5 = 4 \cdot z + 5$ and $t_6 = 9$ then:

$$Tr(\phi) = \forall (x_{t_1}) \exists (x_{t_2}, x_{t_3}) \big( (p_{x,y}((x_{t_1}), (x_{t_2}, x_{t_3})) \vee p_y((x_{t_2}, x_{t_3}), x_{t_4})) \wedge q'(x_{t_5}, x_{t_6}) \big).$$

We consider a new set of kinds $K' = \{k_x, k_y, k_{t_4}, k_{t_5}, k_{t_6}\}$ and a $K'$-kinded logical signature $(\mathcal{B}_2, \Sigma'_L)$ such that $\Sigma'_L$ contains $p_{x,y}$ of type $k_x k_y$, $p_y$ of type $k_y k_{t_4}$, and $q'$ of type $k_{t_5} k_{t_6}$.

We will associate to the $\Omega_L$-algebra $\mathcal{A}$, a $(\mathcal{B}_2, \Sigma'_L)$-logical structure $\mathcal{S}_\mathcal{A} = (S, \Sigma'^{\mathcal{S}_\mathcal{A}}_L)$ with:

- $S_{k_x} = \{0, 1\}$, $S_{k_y} = \{(a, a + 3) \,|\, a \in \mathbf{N}\}$, $S_{k_{t_4}} = \{0, 1, 2, 3, 4\}$, $S_{k_{t_5}} = \{4 \cdot a + 5 \,|\, a \in \mathbf{N}\}$ and $S_{k_{t_6}} = \{9\}$;
- $p_{x,y}(u, v) = \begin{cases} 1 & , \text{ if } u \text{ is even,} \\ 0 & , \text{ otherwise;} \end{cases}$
- $p_y(u, v) = \begin{cases} 1 & , \text{ if } u = (a, b) \text{ and } b \text{ is even,} \\ 0 & , \text{ otherwise;} \end{cases}$
- $q'(u, v) = \begin{cases} 1 & , \text{ if } u\%3 = 1, \\ 0 & , \text{ otherwise.} \end{cases}$

Now, let $\gamma$ be an assignment of $X$ into $\mathcal{A}$ with $\gamma(z) = 2$. The transformed assignment, $Tr(\gamma)$, is an assignment of $X' = \{(x_{t_1}), (x_{t_2}, x_{t_3}), x_{t_4}, x_{t_5}, x_{t_6}\}$ into $\mathcal{S}_\mathcal{A}$ such that $Tr(\gamma)(x_{t_4}) = 2$, $Tr(\gamma)(x_{t_5}) = 13$ and $Tr(\gamma)(x_{t_6}) = 9$.

We can easily see that the transformation above preserves the value of $\phi$, i.e. $\mathcal{I}_\mathcal{A}(\phi)(\gamma) = \mathcal{I}_{\mathcal{S}_\mathcal{A}}(Tr(\phi))(Tr(\gamma)) = 1$.

**Theorem 3.1** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\phi$ a formula over $\Omega_L$ and some set of variables $X$, and $\gamma \in \Gamma(X, \mathcal{A})$. Then,

$$\mathcal{I}_{\mathcal{A}}(\phi)(\gamma) = \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi))(Tr(\gamma)),$$

where $\mathcal{S}_{\mathcal{A}}$, $Tr(\phi)$ and $Tr(\gamma)$ are defined as above.

**Proof** We proceed by structural induction on the formula $\phi$. The following cases are to be discussed:

- $\phi = p(t_1, \ldots, t_m)$. By definition, $Tr(\phi) = p'(x_{t_1}, \ldots, x_{t_m})$. We have that $\mathcal{I}_{\mathcal{A}}(\phi)(\gamma)$ equals the value of $p$ over $(\mathcal{I}_{\mathcal{A}}(t_1)(\gamma), \ldots, \mathcal{I}_{\mathcal{A}}(t_m)(\gamma))$, and

$$\begin{aligned} \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi))(Tr(\gamma)) &= p'^{\mathcal{S}_{\mathcal{A}}}(Tr(\gamma)(x_{t_1}), \ldots, Tr(\gamma)(x_{t_m})) \\ &= p'^{\mathcal{S}_{\mathcal{A}}}(\mathcal{I}_{\mathcal{A}}(t_1)(\gamma), \ldots, \mathcal{I}_{\mathcal{A}}(t_m)(\gamma)), \end{aligned}$$

  which by the definition of $p'^{\mathcal{S}_{\mathcal{A}}}$ implies $\mathcal{I}_{\mathcal{A}}(\phi)(\gamma) = \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi))(Tr(\gamma))$.

- $\phi = \phi_1 \wedge \phi_2$. Assume that $\phi_1$ and $\phi_2$ satisfy the property. We have that

$$\begin{aligned} \mathcal{I}_{\mathcal{A}}(\phi_1 \wedge \phi_2)(\gamma) &= \mathcal{I}_{\mathcal{A}}(\phi_1)(\gamma) \wedge \mathcal{I}_{\mathcal{A}}(\phi_2)(\gamma) \\ &= \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi_1))(Tr(\gamma)) \wedge \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi_2))(Tr(\gamma)) \\ &= \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi_1) \wedge Tr(\phi_2))(Tr(\gamma)) \\ &= \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi_1 \wedge \phi_2))(Tr(\gamma)). \end{aligned}$$

- $\phi = \phi_1 \vee \phi_2$. This is similar to the previous case.

- $\phi = (\forall v)\phi_1$. Assume that $\phi_1$ satisfies the property. We have that

$$\begin{aligned} \mathcal{I}_{\mathcal{A}}((\forall v)\phi_1)(\gamma) &= \textstyle\bigwedge_{a \in A_k} \mathcal{I}_{\mathcal{A}}(\phi_1)(\gamma[v/a]) \\ &= \textstyle\bigwedge_{a \in A_k} \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi_1))(Tr(\gamma[v/a])). \end{aligned}$$

  Notice that in $Tr(\phi_1)$ the terms $v_1$,..., $v_n$ which contain $v$ are replaced by different variables $x_{v_1}$,..., $x_{v_n}$. We define $\phi_2$ as $Tr(\phi_1)$ but the variables $x_{v_1}$,..., $x_{v_n}$ are replaced by the same variable $(x_{v_1}, \ldots, x_{v_n})$ and the predicates $p'$, for some $p \in \Sigma_L$, appearing in atomic formulas containing $v$, are replaced by $p_v$ (similarly to the 4th step in the definition of $Tr(\phi)$). By the definition of the transformation of formulas and assignments,

$$\begin{aligned} \{Tr(\gamma[v/a]) \mid a \in A_k\} &= \{Tr(\gamma)[(x_{v_1}, \ldots, x_{v_n})/(a_1, \ldots, a_n)] \mid \\ &\quad (a_1, \ldots, a_n) \in S_{k_v}\} \end{aligned}$$

and consequently,

$$\begin{aligned}\mathcal{I}_{\mathcal{A}}(\phi)(\gamma) &= \textstyle\bigwedge_{(a_1,\ldots,a_n)\in S_{k_v}} \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(\phi_2)(Tr(\gamma)[(x_{v_1},\ldots,x_{v_n})/(a_1,\ldots,a_n)]) \\ &= \mathcal{I}_{\mathcal{S}_{\mathcal{A}}}((\forall(x_{v_1},\ldots,x_{v_n})\phi_1)(Tr(\gamma)).\end{aligned}$$

- $\phi = (\exists v)\phi_1$. This is similar to the previous case. □

The theorem above permits us to restate all the preservation results from Section 2.2 in the context of logically extended membership algebras.

**Theorem 3.2** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$ and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^T, \exists^T_a, \forall \mid T \subseteq\uparrow b'\}$, for all $x \geq b$, then:

$$[\phi]^{\mathcal{D}} \geq b \Rightarrow [\phi]^{\mathcal{A}} \geq b'$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Omega_L, X)$. Moreover, if we also have that for any $B' \subseteq B$,

$$\vee B' \geq b \Rightarrow (\exists x \in B')(x \geq b),$$

then

$$[\phi]^{\mathcal{D}} \geq b \Rightarrow [\phi]^{\mathcal{A}} \geq b',$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Omega_L, X)$.

**Proof** Suppose that $[\phi]^{\mathcal{D}} \geq b$. This implies $\mathcal{I}_{\mathcal{D}}(\phi)(\gamma) \geq b$, for any $\gamma \in \Gamma(X, \mathcal{D})$.

Let $Tr(\phi)$ be the transformation on the formula $\phi$ defined as above, $\mathcal{S}_{\mathcal{D}}$ the $(\mathcal{B}, \Sigma'_L)$-logical structure associated to $\mathcal{D}$, $\mathcal{S}_{\mathcal{A}}$ the $(\mathcal{B}, \Sigma'_L)$-logical structure associated to $\mathcal{A}$ and $Tr(\gamma) \in \Gamma(X', \mathcal{S}_{\mathcal{D}})$.

Also, consider the equivalence $\rho'$ on $\mathcal{S}_{\mathcal{A}}$ such that

- $\rho'_{k_t}$ is the restriction of $\rho_{k'}$ to $S_{k_t} \times S_{k_t}$, where $k'$ is the kind of a maximal term $t$ appearing in $\phi$;
- if $v$ is a quantified variable and $v_1$,...,$v_n$ are the maximal terms of $\phi$ containing $v$ then, $\rho'_{k_v}$ is defined by $(a_1, \ldots, a_n)\, \rho'_{k_v}\, (b_1, \ldots, b_n)$, if $a_i\, \rho_{k_i}\, b_i$, for any $1 \leq i \leq n$, where $k_i$ is the kind of $v_i$.

We can easily prove that $\mathcal{S}_{\mathcal{D}}$ is an $\alpha$-abstraction of $\mathcal{S}_{\mathcal{A}}$ by $\rho'$. Consequently,

$$\begin{array}{rcl} \mathcal{I}_{\mathcal{D}}(\phi)(\gamma) \geq b & \Rightarrow & \mathcal{I}_{\mathcal{S}_{\mathcal{D}}}(Tr(\phi))(Tr(\gamma)) \geq b \\ & \Rightarrow & (\forall \gamma' \in Tr(\gamma))(\mathcal{I}_{\mathcal{S}_{\mathcal{A}}}(Tr(\phi))(\gamma') \geq b') \\ & \Rightarrow & (\forall \gamma'' \in \gamma)(\mathcal{I}_{\mathcal{A}}(\phi)(\gamma'') \geq b'). \end{array}$$

Above, we have used Theorem 2.2 and

$$\{\gamma' \in \Gamma(X', \mathcal{S}_{\mathcal{A}}) | \gamma' \in Tr(\gamma)\} = \{Tr(\gamma'') | \gamma'' \in \Gamma(X, \mathcal{A}) \text{ and } \gamma'' \in \gamma\}.$$

Since we can extend Lemma 2.1 to logically extended algebras, i.e. for any $\gamma'' \in \Gamma(X, \mathcal{A})$ there exists an assignment $\gamma \in \Gamma(X, \mathcal{D})$ such that $\gamma'' \in \gamma$, we obtain $[\phi]^{\mathcal{A}} \geq b'$. □

**Theorem 3.3** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$ and $b$ a truth value in $B$. If there exists $b' \in B$ such that $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\downarrow b'\}$, for all $x \leq b$, then:

$$[\phi]^{\mathcal{D}} \leq b \Rightarrow [\phi]^{\mathcal{A}} \leq b',$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Omega_L, X)$. Moreover, if

$$\wedge B' \leq b \Rightarrow (\exists x \in B')(x \leq b), \text{ for any } B' \subseteq B,$$

then

$$[\phi]^{\mathcal{D}} \leq b \Rightarrow [\phi]^{\mathcal{A}} \leq b',$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Omega_L, X)$.

**Proof** Similarly to the proof of Theorem 3.2. □

**Theorem 3.4** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$ and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq\uparrow b \cap \downarrow x\}$, for all $x > b$, then:

$$[\phi]^{\mathcal{D}} \geq b \Rightarrow b \leq [\phi]^{\mathcal{A}} \leq [\phi]^{\mathcal{D}} \text{ and}$$
$$[\phi]^{\mathcal{D}} = b \Rightarrow [\phi]^{\mathcal{A}} = b,$$

for any $\phi \in \mathcal{L}^{\{\wedge\}}(\Omega_L, X)$.

**Proof** Similarly to the proof of Theorem 3.2. □

**Theorem 3.5** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$ and $b$ a truth value in $B$. If $\alpha(b) = \forall$ and $\alpha(x) \in \{\exists^S, \forall \mid S \subseteq\uparrow x \cap \downarrow b\}$, for all $x < b$, then:

$$[\phi]^{\mathcal{D}} \leq b \Rightarrow [\phi]^{\mathcal{D}} \leq [\phi]^{\mathcal{A}} \leq b \text{ and}$$
$$[\phi]^{\mathcal{D}} = b \Rightarrow [\phi]^{\mathcal{A}} = b,$$

for any $\phi \in \mathcal{L}^{\{\vee\}}(\Omega_L, X)$.

**Proof** Similarly to the proof of Theorem 3.2. □

**Theorem 3.6** Let $\Omega_L = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}$, $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$ and $b$ a truth value in $B$. If:

1. $\alpha(b) = \forall$;
2. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\uparrow b\}$, for all $x > b$;
3. $\alpha(x) \in \{\exists^{S'}, \exists_a^{S'}, \forall \mid S' \subseteq\downarrow b\}$, for all $x < b$;
4. for any $B' \subseteq B$, $\wedge B' \leq b$ implies that there exists $x \in B'$ such that $x \leq b$;
5. for any $B' \subseteq B$, $\vee B' \geq b$ implies that there exists $x \in B'$ such that $x \geq b$;
6. for any $B' \subseteq B$, $\wedge B' = b$ implies $b \in B'$;
7. for any $B' \subseteq B$, $\vee B' = b$ implies $b \in B'$;

then

$$[\phi]^{\mathcal{D}} \geq b \Rightarrow [\phi]^{\mathcal{A}} \geq b,$$
$$[\phi]^{\mathcal{D}} \leq b \Rightarrow [\phi]^{\mathcal{A}} \leq b,$$
$$[\phi]^{\mathcal{D}} = b \Rightarrow [\phi]^{\mathcal{A}} = b,$$

for any $\phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Omega_L, X)$.

**Proof** Similarly to the proof of Theorem 3.2. □

All preservation results in [111] (for Kleene's 3-valued logic) are particular cases of Theorem 3.6 (we denote by $\mathcal{B}_3$ the corresponding truth algebra).

**Corollary 3.1** Let $\Omega_L = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ be a $K$-kinded logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\rho$ a congruence on $\mathcal{A}$, $\alpha$ an interpretation policy over $\mathcal{B}_3$, and $\mathcal{D} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$. Then, the following properties hold:

- if $\alpha(0) = \forall$, $\alpha(\bot) = \exists^{\{0,\bot,1\}}$, and $\alpha(1) = \exists^{\{0,1\}}$ then

$$[\phi]^{\mathcal{D}} = 0 \;\Rightarrow\; [\phi]^{\mathcal{A}} = 0, \text{ for any } \phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X).$$

- if $\alpha(0) = \exists^{\{0,\bot,1\}}$, $\alpha(\bot) = \exists^{\{\bot,1\}}$, and $\alpha(1) = \forall$, then

$$[\phi]^{\mathcal{D}} = 1 \;\Rightarrow\; [\phi]^{\mathcal{A}} = 1, \text{ for any } \phi \in \mathcal{L}^{\{\wedge,\vee\}}(\Sigma_L, X).$$

- if $\alpha(0) = \forall$, $\alpha(\bot) = \exists_a^{\{0,\bot,1\}}$, and $\alpha(1) = \forall$, then

$$[\phi]^{\mathcal{D}} = b \;\Rightarrow\; [\phi]^{\mathcal{A}} = b, \text{ for any } \phi \in \mathcal{L}(\Sigma_L, X) \text{ and } b \in \{0,1\}.$$

## 3.4 Particular cases

Following [111] we prove that many abstraction techniques from the literature are particular cases of ours.

### 3.4.1 McMillan's Approach

In [86], McMillan has proposed a kind of data type reduction (abstraction) to be used with the SMV system. Even though his approach was "susceptible" to be a particular case of Cousot's abstract interpretation (see [86]), McMillan has preferred to develop this new formalism closer to data types and practical applications. This is not an isolated case and it proves the necessity of an intermediate formalism with a high degree of generality but easy to be applied in practice.

Our approach, proposed in the previous sections, is as simply and elegant as general it is; it can be used to handle abstractions of data types defined in the most general way. This subsection, as well as the next three, will prove this.

**McMillan's Approach** We will use mainly the same notation as in [86]. Let $U$ be a set of values, $V$ a set of variables, $T$ a set of types, and $C$ a set of constructors (each of them having an arity). Define the set $\mathcal{L}$ of formulas as being the set of ground terms over $C$.

A *structure* is a triple $M = (\mathcal{R}, \mathcal{N}, \mathcal{F})$, where $\mathcal{R} : T \to \mathcal{P}(U)$ assigns a range of values to every type, $\mathcal{N}$ is a set of denotations, and $\mathcal{F}$ is an interpretation assigning a function $\mathcal{F}(c) : \mathcal{N}^{n_c} \to \mathcal{N}$ to each constructor $c$ of arity $n_c$. The denotation of a formula $\phi \in \mathcal{L}$ in a structure $M$ is defined inductively and denoted by $\phi^M$. It is assumed that each structure $M$ admits a pre-order $\leq$ on $\mathcal{N}$ and $\mathcal{F}(c)$ is a monotonic function with respect to $\leq$, for all constructors $c$.

A *homomorphism* from a structure $M = (\mathcal{R}, \mathcal{N}, \mathcal{F})$ into a structure $M' = (\mathcal{R}', \mathcal{N}', \mathcal{F}')$ is a function $h : \mathcal{N} \to \mathcal{N}'$ satisfying

$$\mathcal{F}'(c)(h(t_1), \ldots, h(t_n)) \leq' h(\mathcal{F}(c)(t_1, \ldots, t_n)),$$

for all $c(t_1, \ldots, t_n) \in \mathcal{L}$ ($\leq'$ is the pre-order on $\mathcal{N}'$).

By structural induction and using the monotonicity we can easily obtain $\phi^{M'} \leq' h(\phi^M)$, for all $\phi \in \mathcal{L}$ and homomorphisms $h$ from $M$ into $M'$.

Assume now that each structure $M$ contains a distinguished denotation *true* (the denotation of valid formulas). A homomorphism $h$ from $M$ into $M'$ is *truth preserving* if $true' \leq' h(x)$ implies $x = true$. If $h$ is truth preserving, then

$$(*) \quad \phi^{M'} = true' \quad \Rightarrow \quad \phi^M = true,$$

for all formulas $\phi$.

The methodology described above can be used in connection with data type reductions as follows. Let us assume that $M'$ is obtained from $M$ by some abstraction technique. If there is a truth preserving homomorphism $h$ from $M$ into $M'$, then $(*)$ holds true. Therefore, proving that a formula $\phi$ holds in $M$ can be reduced to proving that $\phi$ holds in $M'$. This task could be easier because $M'$ is a reduced structure obtained from $M$.

The abstraction technique considered in [86] works as follows. Let $M = (\mathcal{R}, \mathcal{N}, \mathcal{F})$ be a structure, where

$$\mathcal{N} = \{\mathcal{I} | \mathcal{I} : \{\gamma | \gamma : V \to U\} \to U\},$$

and let $r : T \to \mathcal{P}(U)$ be a function such that $r(s) \subseteq \mathcal{R}(s)$, for all $s \in T$. Define

$$\mathcal{R}_r(s) = \{\{a\} | a \in r(s)\} \cup \{\mathcal{R}(s) - r(s)\}$$

and $\mathcal{N}_r$ as $\mathcal{N}$ but changing $U$ into $\mathcal{P}(U)$. The interpretation $\mathcal{F}_r$ is not defined in the general case, but only in the case of the SMV operators, and it is asked that every constructor $c$ should be *safe with respect to* $r$, that is

$$\mathcal{F}(c(t_1, \ldots, t_n))(\gamma) \in \mathcal{F}_r(c(t'_1, \ldots, t'_n))(\gamma'),$$

for all $t_1, t'_1, \ldots, t_n, t'_n$, $\gamma : V \to U$ and $\gamma' : V \to \mathcal{P}(U)$ such that $\gamma \in \gamma'$ and $\mathcal{F}(t_i)(\gamma) \in \mathcal{F}_r(t'_i)(\gamma')$, for all $1 \leq i \leq n$.

Then, $h_r : M \to M_r$ given by

$$h_r(x)(\gamma') = \{x(\gamma) | \gamma \in \gamma'\},$$

for all $x \in \mathcal{N}$ and $\gamma' : V \to \mathcal{P}(U)$, is a truth preserving homomorphism from $M$ into $M_r$. Therefore, $(*)$ can be applied.

**A Membership Algebra Approach** The approach presented above is a particular case of the results in Section 3.3. The set of types is just a set $K$ of kinds and the truth algebra under consideration is the classical one with only two values, denoted $\mathcal{B}_2$. The set of constructors used to define formulas is in our case the logically extended membership signature $\Omega_L = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L, \pi)$. The set $\mathcal{L}(\Omega_L, X)$ of formulas we consider is incomparable much larger than the set of formulas considered by McMillan. Moreover, we consider membership signatures as in Definition 3.1 to define concrete (and arbitrarily complex) data types. A structure is then an $\Omega_L$-algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ which induces both assignments and formula interpretations.

A data type reduction $r$ (in McMillan's approach) induces a $K$-kinded equivalence $\rho = (\rho_k | k \in K)$ given by

$$a \, \rho_k \, b \quad \Leftrightarrow \quad (a = b \in r(k)) \vee (a, b \notin r(k)),$$

for all $a, b \in A_k$.

This equivalence should be in fact a congruence (this it is not mentioned explicitly in [86] but it can be easily seen from examples).

Since the homomorphism $h$ from $M$ into $M_r$ is truth-preserving, the structure $M_r$ becomes now an $\alpha$-abstraction of $\mathcal{A}$ by $\rho$, where $\alpha(0) = \exists$ and $\alpha(1) = \forall$. The definition we consider for the $\alpha$-abstraction makes safe with respect to $r$ all the function and logical symbols. Notice that there is no need to consider pre-orders on algebras.

Therefore, McMillan's approach is a particular case of our approach.

### 3.4.2 Shape Analysis

*Shape analysis* is a data flow analysis technique [96]. It is mainly used for complex analysis of dynamically allocated data, and it is based on representing the set of possible memory states ("stores") that arise at a given point in the program by *shape graphs*. In such a graph, heap cells are represented by shape-graph nodes and, in particular, sets of "indistinguishable" heap cells are represented by a single shape-graph node. In the past two decades, many

shape-analysis algorithms have been developed [69, 70, 77, 67, 15, 110, 100, 114, 103]. The parametric framework for shape analysis introduced in [104] covers almost all of the work mentioned above.

We show that shape analysis as described in [104] can be obtained as a particular case of the results in Section 3.3.

**Shape Analysis** In [104], the authors define a first order logic with transitive closure over a finite set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of predicate symbols. A *2-valued logical structure* for this logic is defined as a couple $S = (U^S, \mathcal{I}^S)$, where $U^S$ is a set of individuals and $\mathcal{I}^S$ maps each predicate symbol $p$ of arity $k$ to a truth-valued function $\mathcal{I}^S(p) : (U^S)^k \to \{0, 1\}$. Replacing the set $\{0, 1\}$ by the set $\{0, 1, \bot\}$ in the definition above we obtain a *3-valued logical structure*.

*2-valued logical structures* are used to encode *concrete stores* as follows. Individuals represent memory locations in the heap, pointers from the stack into the heap are represented by unary predicates, and pointer valued-fields are represented by binary predicates. The *property-extraction principle* adopted in [104] is the following: by encoding stores as logical structures, questions about properties of stores can be answered by evaluating formulas. *3-valued logical structures* are used to encode *abstract stores*.

The concrete store is related to the abstract store by *truth-blurring embeddings*. An *embedding* of a structure $S = (U^S, \mathcal{I}^S)$ into a structure $S' = (U^{S'}, \mathcal{I}^{S'})$ is any surjective function $f : U^S \to U^{S'}$ such that

$$\mathcal{I}^S(p)(u_1, \ldots, u_k) \sqsubseteq \mathcal{I}^{S'}(p)(f(u_1), \ldots, f(u_k)),$$

for any $k$, predicate symbol $p$ of arity $k$, and any $u_1, \ldots, u_k \in U^S$, where $\sqsubseteq$ is the information order on $\{0, 1, \bot\}$ defined by

$$x \sqsubseteq y \Leftrightarrow x = y \vee y = \bot.$$

**Theorem 3.7** *(Embedding Theorem [104])*
*Let $f$ be an embedding from a logical structure $S = (U^S, \mathcal{I}^S)$ into a logical structure $S' = (U^{S'}, \mathcal{I}^{S'})$. Then,*

$$\mathcal{I}^S(\varphi)(\gamma) \sqsubseteq \mathcal{I}^{S'}(\varphi)(f \circ \gamma),$$

*for any formula $\varphi$ and any complete assignment $\gamma$ for $\varphi$.*

The embedding theorem provides a systematic way to use an abstract 3-valued structure $S$ to answer questions about properties of the concrete 2-valued structure that $S$ represents. It ensures that it is safe to evaluate a formula $\varphi$ on a single 3-valued structure $S$ instead of evaluating $\varphi$ in all 2-valued structures that can be embedded in $S$.

**A Membership Algebra Approach** The approach in [104] can be easily seen as a particular case of the approach proposed in this paper. For each logical structure $S = (U^S, \mathcal{I}^S)$, $U^S$ can be viewed as a uni-kinded $\Omega_L$-algebra, where the truth algebra is the one used for Kleene's three-valued logic, denoted $\mathcal{B}_3$, $\Omega_L = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ and $\Sigma$ contains only the operators $(\_, \ldots, \_)$. $\mathcal{I}^S$ is the interpretation function of formulas into the algebra $U^S$ (see Section 3.2).

The abstraction is driven by embeddings which are surjective functions. As we have no real function symbols, the equivalence relation induced by such a surjective function is a congruence. Also, the properties an embedding satisfy give rise to $\alpha$-abstractions with $\alpha(0) = \alpha(1) = \forall$ and $\alpha(\bot) = \exists_a$, and the embedding theorem follows easily from the results in Section 3.3. Moreover, in contrast to [104], the set of individuals in our approach is typed and enriched by typed-operations; the predicate symbols are typed as well.

### 3.4.3 Predicate Abstraction

*Predicate abstraction*, also called *boolean abstraction* or *existential abstraction* in [4], has been introduced by Graf and Saidi in [57] to provide a method for the automatic construction of an abstract state graph of an infinite system using the PVS theorem prover. Since then, predicate abstraction has been studied thoroughly [36, 113].

The main idea of the predicate abstraction is to map concrete objects (states of a transition system, data of a data type etc.) to "abstract objects" according to their evaluation under a finite set of predicates.

Let $P$ be a finite set of predicates over a set $A$. The set $P$ induces an equivalence relation $\rho_P$ on $A$ as follows

$$a \ \rho_P \ b \quad \Leftrightarrow \quad (\forall p \in P)(p(a) \text{ iff } p(b)),$$

for all $a, b \in A$.

The quotient set $A/\rho_P$ can be taken as the abstract system induced by $P$. If some operations are given on the set $A$ (i.e., a transition relation, data type operations etc.) then these operations should be redefined to operate on the abstract system in a congruential way.

It is very clear that predicate abstraction is a particular case of the approach proposed in this paper.

### 3.4.4 Duplicating Predicate Symbols

The technique of *duplicating predicate symbols* is one of the intensively used techniques in abstraction [27, 35, 6]. It is based on associating "copies" to

each predicate symbol. Therefore, a formula $\varphi$ gets several versions depending on the predicate copies are used. Usually, two copies for each predicate symbol are associated, and two versions of a formula are used: one of them for validation, and the other one for refutation.

We describe in details the duplicating predicate symbols technique used in [6] and we will show that it can be easily obtained as a particular case of the abstraction methodology proposed in this paper.

In [6], Bidoit and Boisseau considered logically extended structures (we use here logically extended membership algebras) whose predicate symbols are valuated into $\{0, 1\}$, and associated a *split signature* $\Omega_L^S = (\mathcal{B}_2, \leq', \Sigma, \Sigma_{L,\oplus} \cup \Sigma_{L,\ominus}, \pi')$ to each logically extended signature $\Omega_L = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L, \pi)$, where $0 \leq' 1$, and $\Sigma_{L,\oplus}$ and $\Sigma_{L,\ominus}$ are obtained by indexing the predicate symbols $P \in \Sigma_L$ by $\oplus$ and, respectively, by $\ominus$. $P_\oplus$ and $P_\ominus$ have the same type as $P$.

Let $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ and $\mathcal{C} = (C, \Sigma^C, \Pi_C)$ be two $\Omega_L$-algebras and $h$ be an epimorphism from $\mathcal{A}$ into $\mathcal{C}$. The *canonical* $\Omega_L^S$*-structure* associated to $\mathcal{A}$ and $h$ is defined by $\mathcal{A}^h = (C, \Sigma^C, \Pi'_C)$ such that:

- $(b_1, \ldots, b_n) \in \Pi'_C(s_{P_{\oplus,1}})$ if $(\forall 1 \leq i \leq n)(\forall a_i \in h^{-1}(b_i))((a_1, \ldots, a_n) \in \Pi_A(s_{P,1})$;
- $(b_1, \ldots, b_n) \in \Pi'_C(s_{P_{\ominus,1}})$ if $(\forall 1 \leq i \leq n)(\exists a_i \in h^{-1}(b_i))((a_1, \ldots, a_n) \in \Pi_A(s_{P,1})$;

for all $P \in \Sigma_L$ of logical type $k_1 \cdots k_n$ and $b_1 \in C_{k_1}, \ldots, b_n \in C_{k_n}$. Also, $\Pi'_C(s) = \Pi_C(s)$, for any sort $s$ that does not represent a predicate.

For any formula $\varphi$ over $\Omega_L$ and $X$ define two formulas $\varphi_\oplus$ and $\varphi_\ominus$ over $\Omega_L^S$ and $X$ as follows:

- $P(t_1, \ldots, t_n)_\oplus = P_\oplus(t_1, \ldots, t_n)$;
- $P(t_1, \ldots, t_n)_\ominus = P_\ominus(t_1, \ldots, t_n)$;
- $(\varphi_1 \vee \varphi_2)_\oplus = (\varphi_{1\oplus} \vee \varphi_{2\oplus})$ and $(\varphi_1 \vee \varphi_2)_\ominus = (\varphi_{1\ominus} \vee \varphi_{2\ominus})$, and similar for $\wedge$, $\forall$ and $\exists$;
- $(\neg\varphi)_\oplus = \neg(\varphi_\ominus)$ and $(\neg\varphi)_\ominus = \neg(\varphi_\oplus)$.

Now, one of the main results proved in [6] states that

$$[\varphi_\oplus]^{\mathcal{A}^h} = 1 \quad \Rightarrow \quad [\varphi]^{\mathcal{A}} = 1$$

and

$$[\varphi_\ominus]^{\mathcal{A}^h} = 0 \quad \Rightarrow \quad [\varphi]^{\mathcal{A}} = 0.$$

These two implications give the correctness of the abstraction. Here, the abstraction should be understood by the fact that an element $b$ acts as an abstraction for $h^{-1}(b)$.

In order to show that the result above is a particular case of the abstraction methodology proposed in this paper we have to remark first the following:

- the predicate symbols, in the formalism above, are valuated into $\{0, 1\}$;
- the kernel $ker(h)$ of any epimorphism $h$ defines a congruence on $\mathcal{A}$ and the quotient algebra $\mathcal{A}/ker(h)$ is isomorphic to $\mathcal{C}$. Conversely, any congruence $\rho$ on $\mathcal{A}$ leads to an epimorphism from $\mathcal{A}$ into $\mathcal{A}/\rho$. Therefore, we can consider congruences instead of epimorphisms in order to design abstractions;
- any formula $\varphi \in \mathcal{L}(\Omega_L, X)$ can be equivalently transformed into the *negation normal form*, where the negation is applied only to atomic formulas.

Given a logically extended signature $\Omega_L = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L, \pi)$ we define a new signature $\Omega'_L = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L \cup \Sigma'_L, \pi)$, where $\Sigma'_L$ is just a copy of $\Sigma_L$. In any $\Omega'_L$-algebra $\mathcal{A}$, the predicate symbol $P'$ will be interpreted as $\neg P$ is.

For any formula $\varphi \in \mathcal{L}(\Omega'_L, X)$ in the negation normal form we define a new formula $\varphi' \in \mathcal{L}^{\{\wedge,\vee\}}(\Omega'_L, X)$ by replacing "$\neg P$" by "$P'$" and "$\neg Q'$" by "$Q$", for any $P$ and $Q$.

Directly from the above constructions we have

$$(*) \quad [\varphi]^{\mathcal{A}} = [\varphi']^{\mathcal{A}}, \text{ for any formula } \varphi \in \mathcal{L}(\Omega'_L, X).$$

Consequently,

- if $\mathcal{D}$ is an $\alpha_1$-abstraction of $\mathcal{A}$ by $\rho$, with $\alpha_1(1) = \forall$ and $\alpha_1(0) = \exists$, then
$$[\varphi']^{\mathcal{D}} = 1 \quad \Rightarrow \quad [\varphi]^{\mathcal{A}} = 1$$
(from Theorem 3.6 and $(*)$);
- if $\mathcal{D}$ is an $\alpha_2$-abstraction of $\mathcal{A}$ by $\rho$, with $\alpha_2(1) = \exists$ and $\alpha_2(0) = \forall$, then
$$[\varphi']^{\mathcal{D}} = 0 \quad \Rightarrow \quad [\varphi]^{\mathcal{A}} = 0$$
(from Theorem 3.6 and $(*)$).

$\varphi'$ plays exactly the role of $\varphi_{\oplus}$ in the first case (of $\alpha_1$-abstractions), and it plays exactly the role of $\varphi_{\ominus}$ in the second case (of $\alpha_2$-abstractions). In both cases, $\mathcal{D}$ plays the role of $\mathcal{A}^h$.

These results emphasize clearly the nature of the duplicating predicate symbols technique. Thus, this technique consists of two abstractions based on the same congruence. One of them is an $\alpha_1$-abstraction, used for validation, and the other one is an $\alpha_2$-abstraction, used for refutation.

## 3.5 Abstractions of Abstract Data Types

In this section we generalize the results from the previous sections to abstract data types. An *abstract data type* (ADT, for short) for a membership signature $\Omega_L$ is a class of $\Omega_L$-algebras that is closed under isomorphism [2]. An ADT is called *monomorphic* if its algebras are all isomorphic to each other; otherwise, it is called *polymorphic* [3]. A *specification* may be viewed as a description of a class of objects by means of their properties. In a formal specification these properties are expressed as formulas in a logic. Hence, a specification of an ADT essentially consists of a set of formulas expressing the common properties of its algebras. Specifications are defined by a syntax and a semantics. The syntax fixes the "form", and the semantics fixes the "meaning" of specifications. A specification is called monomorphic (polymorphic) if it defines a monomorphic (polymorphic) ADT. Specifications can be classified in *atomic* and *composed*. An atomic specification is essentially built up from the scratch; it consists of a signature $\Omega_L$ and a set of formulas $\Phi$ in a logic $L$. Its semantics is defined as the class of all $\Omega_L$-algebras that are models of $\Phi$. Three basic atomic specifications are *loose specification*, *initial specification*, and *constructive specification* [4]. A composed specification is a specification written in a specification language. Starting from atomic specifications the constructs of such a language allow one to build large specifications out of smaller ones [5].

[2]Informally, the closure under isomorphism corresponds to the fact that isomorphic algebras are "similar" in that they differ only by the nature of their carriers.

[3]A monomorphic ADT stands for a single data type, whereas a polymorphic ADT may correspond to an "incomplete" specification.

[4]Drawing up atomic specifications is sometimes called *specification-in-the-small*.

[5]Drawing up composed specifications with the help of a specification language is sometimes called *specification-in-the-large*.

### 3.5.1 Abstractions of Initial Specifications

An *initial membership specification* is a 4-tuple $Sp = (\Sigma, \pi, X, E)$, where $\Omega = (\Sigma, \pi)$ is a membership signature, $X$ is a disjoint family of variables, disjoint of $\Sigma$ too, and $E$ is a set of sentences over $\Omega$ and $X$.

The *semantics* of an initial membership specification $Sp$ called the monomorphic abstract data type (ADT, for short) induced by $Sp$, is defined by

$$\mathcal{M}(Sp) = \{\mathcal{A} | \mathcal{A} \text{ is isomorphic to } \mathcal{T}_{\Omega,E}\}.$$

Specifications of logically extended membership algebras are simply membership specifications where sentences for predicate symbols are separately given.

**Definition 3.5** A *logically extended membership specification* is a tuple $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$, where:

1. $(\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ is a logically extended membership signature;
2. $X$ is a disjoint family of variables, disjoint of $\Sigma$ too;
3. $E$ is a set of sentences over $X$ and the membership signature $\Omega = (\Sigma, \pi)$ that does not contain the operators $(\_, \ldots, \_)$ and the sorts $s_{p,b}$;
4. $E_P$ is a set of sentences of the form:
   (a) $t : s_{p,b}$ `if` $\mathcal{C}$;
   (b) $x : s_{p,b_1}$ `if` $x : s_{p,b_2}$,

   where $p \in \Sigma_{L,w}$, $t$ is a term of kind $k_w$ over the membership signature $\Omega$ and $X$, $x$ a variable of kind $k_w$, $b, b_1, b_2 \in B - \{0'\}$, $b_1 \prec' b_2$ and $\mathcal{C}$ is a set of atomic formulas over $\Omega$ and $X$. $E_P$ contains exactly one sentence of type $(4b)$ for every $p \in \Sigma_L$ and $b_1, b_2$ with $b_1 \prec' b_2$.

   Moreover, $E_P$ should satisfy the following *consistency requirement*: for any predicate $p \in \Sigma_L$ and $b_1, b_2 \in B - \{0'\}$ such that there exist $x \in B$ with $x \prec' b_1$ and $x \prec' b_2$, there exist no term $t$ with $E \cup E_P \vdash t : s_{p,b_1}$ and $E \cup E_P \vdash t : s_{p,b_2}$.

**Definition 3.6** Let $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ be a logically extended membership specification. The *initial semantics* of $LSp$, $\mathcal{M}(LSp)$, is the class of all $\Omega_L$-algebras isomorphic to $\mathcal{T}_{\Omega_L, E \cup E_P}$.

Since logically extended membership specifications are usual membership specifications, we have:

**Proposition 3.1** Let $LSp = (\mathcal{B}, \Sigma, \Sigma_L, \pi, X, E, E_P)$ be a logically extended membership specification. Then, $\mathcal{T}_{\Omega_L, E \cup E_P}$ is an initial algebra in the initial semantics of $LSp$.

In the following, by the value of a predicate $p$ over a term $t$ we mean the maximum truth value $b \in B$ (with respect to $\leq'$) for which $E \cup E_P \vdash t : s_{p,b}$, or $0'$, if such a $b$ does not exist.

An abstraction of an $\Omega_L$-algebra $\mathcal{A}$ consists in a congruence $\rho$ and an interpretation policy used to redefine predicate symbols in the quotient of $\mathcal{A}$ with respect to $\rho$. Naturally, abstractions can be applied to abstract data types $\mathcal{M}(LSp)$ by means of a representative of them, and $T_{\Omega, E \cup E_P}$ is a suitable choice.

When an abstraction is specified by a set of equations we say that it is an *equationally specified abstraction*.

**Definition 3.7** Let $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ be a logically extended membership specification. An abstraction of $LSp$ is a pair $\Delta = (A, \alpha)$, where $A$ is a set of sentences over $\Omega$ and $X$ and $\alpha$ is an interpretation policy over $\mathcal{B}$.

**Definition 3.8** Let $LSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_P)$ be a logically extended membership specification and $\Delta = (A, \alpha)$ an abstraction of it. The *logically extended membership specification for the abstraction of $LSp$ by $\Delta$* is $LSp_\Delta = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E_P^\Delta)$, where:

- $E \cup A \cup E_P^\Delta \vdash t : s_{p,b}$ iff the value of $p$ over the set of terms $\{t' | E \cup A \vdash t = t'\}$ according to $\alpha$ is greater than or equal to $b$ (with respect to $\leq'$).

Now, from the definition of $LSp_\Delta$ we can derive the following result which states that the initial semantics of $LSp_\Delta$ contains algebras that are $\alpha$-abstractions of the algebras from the initial semantics of $LSp$.

**Proposition 3.2** Let $LSp$, $\Delta$ and $LSp_\Delta$ be as above. Let $\rho$ be the congruence on $\mathcal{T}_{\Omega, E'}$ defined by:

$$[t]_{=_{E',k}} \ \rho_k \ [t']_{=_{E',k}} \Leftrightarrow E'' \vdash t = t', \text{ for any } t, t' \in (T_{\Omega_L})_k,$$

where $E' = E \cup E_P$ and $E'' = E \cup A \cup E_P^\Delta$. Then, $\mathcal{T}_{\Omega_L, E''}$ is the $\alpha$-abstraction of $\mathcal{T}_{\Omega_L, E'}$ by $\rho$.

In order to extend the results regarding the computation of multi-valued abstractions using 2-valued abstractions to the case of abstract data types, notice that $S'$-congruences, $(\rho, S')$, of algebras from the semantics of a membership specification can be captured by pairs $(A, S')$ where $A$ is a set of

membership sentences. The specification for the quotient abstract data type is obtained as follows.

**Definition 3.9** Let $LSp$ be as above and $\Delta' = (A, S')$ an $S'$-congruence of it. The *logically extended membership specification for the quotient of $LSp$ by $\Delta'$* is $LSp_{\Delta'} = (B, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E_P^{\Delta'})$, where:

- $E'' \vdash t : s \Leftrightarrow (\forall t')(E'' \vdash t = t' \Rightarrow E' \vdash t' : s)$, for any $s \in S'$ and $t, t' \in (T_{\Omega_L, E'})_{\pi(s)}$ ($E' = E \cup E_P$ and $E'' = E \cup A \cup E_P^{\Delta'}$).

Moreover, we can prove that the initial semantics of $LSp_{\Delta'}$ contains algebras that are quotients by an $S'$-congruence of the algebras from the initial semantics of $LSp$. This guarantees that we can relate $\alpha$-abstractions to quotients by an $S'$-congruence as in Section 3.3.1.

**Proposition 3.3** Let $LSp$, $\Delta'$ and $LSp_{\Delta'}$ be as above. Let $\rho$ be the congruence on $\mathcal{T}_{\Omega, E'}$ defined by:

$$[t]_{=_{E',k}} \; \rho_k \; [t']_{=_{E',k}} \Leftrightarrow E'' \vdash t = t', \text{ for any } t, t' \in (T_{\Omega_L})_k,$$

where $E' = E \cup E_P$ and $E'' = E \cup A \cup E_P^{\Delta'}$. Then, $\mathcal{T}_{\Omega_L, E''}$ is the quotient of $\mathcal{T}_{\Omega_L, E'}$ by $(\rho, S')$.

**Example 3.4** A logically extended specification of an elementary data type of natural numbers is given, where the predicate $Isgrz$ is like in Example 3.1 (the logic is under the 2-valued interpretation).

| | |
|---|---|
| **LSpec:** | Natural numbers |
| **kinds:** | `nat` |
| **sorts:** | $s_{Isgrz,1}$ of kind `nat` |
| **opns:** | $0 : \rightarrow$ `nat` |
| | $Succ$ : `nat` $\rightarrow$ `nat` |
| | $Add$ : `nat nat` $\rightarrow$ `nat` |
| **vars:** | $x, y$ : `nat` |
| **E**: | $Add(x, 0) = x$ |
| | $Add(x, Succ(y)) = Succ(Add(x, y))$ |
| **E$_P$:** | $Succ(x) : s_{Isgrz,1}$ |

Let us assume now that we want to prove that the property

$$\varphi = (\forall x, y)(Isgrz(x) \ \vee \ Isgrz(y) \quad \Rightarrow \quad Isgrz(Add(x, y)))$$

holds in the abstract data type defined by *LSpec*. We consider an abstraction $(\Delta, \alpha)$, where $\Delta$ consists in the equation

$$Succ(Succ(x)) = Succ(0),$$

$\alpha(0) = \exists^{\{0,1\}}$, and $\alpha(1) = \forall$. This abstraction treats the number 0 as an individual, and all the natural numbers greater than 0 on the whole.

The data type for the abstraction has only two elements and it is straightforward to prove that $\varphi$ holds. Since the abstraction is of type $\alpha$, we deduce that $\varphi$ holds true in the concrete data type.

**Example 3.5** Consider the program Keeping-up [82] given in Figure 3.1. It

local $x, y$: integer where $x = y = 0$

$$P_1 :: \left[ \begin{array}{l} l_0 : \ loop\ forever\ do \\ \left[ \begin{array}{l} l_1 : \ await\ x < y + 1 \\ l_2 : \ x := x + 1 \end{array} \right] \end{array} \right] \quad \| \quad P_2 :: \left[ \begin{array}{l} m_0 : \ loop\ forever\ do \\ \left[ \begin{array}{l} m_1 : \ await\ y < x + 1 \\ m_2 : \ y := y + 1 \end{array} \right] \end{array} \right]$$

Figure 3.1: Program Keeping-up

consists of two processes $P_1$ and $P_2$. The process $P_1$ repeatedly increments $x$, provided that $x$ does not exceed $y + 1$. Similarly, the process $P_2$ repeatedly increments $y$, provided that $y$ does not exceed $x + 1$. The program satisfies the global safety property $\Box(|x - y| \leq 1)$ [82].

Below, an initial logically extended specification of this program is given. *Conv* stands for "conversion" of boolean data to natural numbers, *Leq* stands for "less than or equal to", and *Add* for "addition". We use $t : s_{p,\perp,1}$ instead of $t : s_{p,\perp}$ and $t : s_{p,1}$, for any predicate symbol $p$.

**LSpec':** Keeping-up

**kinds:** `nat`, `couple`, `bool`
**sorts:** $s_{R,\perp}$, $s_{R,1}, s_{GS,\perp}$, $s_{GS,1}$ of kind `couple`
**opns:** $Zero : \rightarrow$ `nat`
$True : \rightarrow$ `bool`
$False : \rightarrow$ `bool`
$Succ$ : `nat` $\rightarrow$ `nat`
$Conv$ : `bool` $\rightarrow$ `nat`

$Leq$ : $\texttt{nat}\,\texttt{nat} \to \texttt{bool}$
$Add$ : $\texttt{nat}\,\texttt{nat} \to \texttt{nat}$
$(\_, \_)$ : $\texttt{nat}\,\texttt{nat} \to \texttt{couple}$

**vars:** $x, y$ : $nat$

**E:**
$Conv(False) = Zero$
$Conv(True) = Succ(Zero)$
$Add(x, Zero) = x$
$Add(x, Succ(y)) = Succ(Add(x, y))$
$Leq(Zero, x) = True$
$Leq(Succ(x), Zero) = False$
$Leq(Succ(x), Succ(y)) = Leq(x, y)$

**$\mathbf{E}_P$:**
$(0, 0) : s_{R,\perp,1}$
$(Add(x, Conv(Leq(x, y))), y) : s_{R,\perp,1}$ $\texttt{if}$ $(x, y) : s_{R,\perp,1}$
$(x, Add(y, Conv(Leq(y, x)))) : s_{R,\perp,1}$ $\texttt{if}$ $(x, y) : s_{R,\perp,1}$
$(x, y) : s_{GS,\perp,1}$ $\texttt{if}$ $x = y$
$(x, y) : s_{GS,\perp,1}$ $\texttt{if}$ $x = y + Succ(0)$
$(x, y) : s_{GS,\perp,1}$ $\texttt{if}$ $x + Succ(0) = y$

Two three-valued predicates are used: $R$ which is 1 for the couples $(x, y)$ reachable from $(0, 0)$ by the program in Figure 3.1, and $GS$ which describes "safe" couples. They are used to model the global safety property we want to prove. This property can be described by the formula

$$\varphi = (\forall (x, y))\big((x, y) \text{ reachable } \Rightarrow\ (x = y\ \vee\ x = Succ(y)\ \vee\ y = Succ(x))\big)$$

or, equivalently,

$$\varphi = (\forall (x, y))(R(x, y) \Rightarrow GS(x, y)).$$

By means of the equivalence

$$(\forall x, y)(|Succ(x) - Succ(y)| \leq 1 \quad \Leftrightarrow \quad |x - y| \leq 1)$$

we can derive the abstraction $(\Delta, \alpha)$, where $\Delta$ consists in the equation

$$(Succ(x), Succ(y)) = (x, y),$$

$\alpha(0) = \forall$, $\alpha(\perp) = \exists^{\{0,\perp,1\}}$, and $\alpha(1) = \forall$.

The abstraction leads to three equivalence classes on the set of all reachable vectors from the initial vector $[(0, 0)]$, namely

$$[(0, 0)],\ [(Succ(0), 0)],\ [(0, Succ(0))].$$

Now, it is straightforward to check that $\varphi$ holds true in the abstract system. As the abstraction uses the interpretation policy $\alpha$, by Corollary 3.1, $\varphi$ holds true in the original system.

# Chapter 4

# Abstraction of Dynamic Data Types

We present an abstraction technique for dynamic systems modeled by membership specifications [91] and specifications given in multi-valued $CTL^*$, which extends the approach in [111]. We have used membership specifications because they provide a suitable framework in which a very wide range of total and partial equational specification formalisms can be naturally represented. The membership algebra formalism is quite general and expressive, supports sub-sorts and overloading, and deals very well with errors and partiality. Moreover, membership algebra specifications can be efficiently implemented in systems like Maude [30]. The temporal logic we consider is under a multi-valued interpretation because there are many problems in software engineering, such as modeling systems with uncertain information or inconsistencies, for which 2-valued logic is insufficient.

The main advantages of the abstraction technique we propose in this chapter are:

- we use equationally specified abstractions as in [92, 84, 111];
- the logic used is multi-valued $CTL^*$. Moreover, maximal finite paths are allowed;
- we interpret the transition function of the abstract system and its atomic propositions in various ways getting many property preservation results. The interpretation in [92, 84] falls in one of these cases and, therefore, the approach in [92, 84] becomes a special case of ours;
- the preservation results were first developed on multi-valued Kripke structures and then translated to membership specifications. This fact

gives independence to these results allowing them to be used with various formalisms;

- we do not represent in our specifications the set of truth values by means of a kind as it was done in [92, 84]. The reason is that such a representation may not allow to distinguish between predicate interpretations in the abstract system or may lead to useless abstractions, as it was already remarked in [111].

## 4.1 Dynamic Data Types

There have been proposed several approaches for modeling dynamic systems by universal algebras (see [2] for a survey on this topic). All the approaches are based on predicates which are added somehow to the signature, but they work outside the algebra. In the approach we propose [45] we model dynamic systems (that can be described in terms of states and transitions) by membership algebras, a suitable logical framework in which a very wide range of total and partial equational specification formalisms can be naturally represented [91]. The membership algebra formalism is quite general and expressive, supports sub-sorts and overloading, and deals very well with errors and partiality. Moreover, membership algebra specifications can be efficiently implemented in systems like Maude [30]. We also add predicates to membership algebras as in the approaches mentioned in [2], but they work inside the algebra (including the transition predicate too). This makes the formalism algebra-logic work unitarily.

In order to model dynamic systems by membership algebras we consider logically extended algebras with a special kind `state`, a set of predicates of type `state` (defining properties of states), and a transition predicate of type `state state`.

**Definition 4.1** A *dynamic $K$-kinded signature* is a $K$-kinded logically extended signature $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$, where:

- the set of basic kinds $K'$ contains one distinguished kind `state`;
- $\Sigma_{L,w} = \emptyset$, for any $w \in K'^{+} - \{\texttt{state}, \texttt{state state}\}$ and $\Sigma_{L,\texttt{state state}} = \{\rightarrow\}$. We have only one predicate $\rightarrow$ of type `state state` called *the transition predicate* and some predicates of type `state`, called *atomic propositions*.

The kind associated to the logical type `state state` is denoted `step`.

As a dynamic signature is a particular case of logically extended signature, it should be thought as an ordinary membership signature that contains two distinguished kinds and some distinguished sorts; the predicate symbols are just used to specify the distinguished sorts.

**Definition 4.2** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic $K$-kinded signature. A *dynamic $\Omega_D$-algebra* $\mathcal{A}$ is an $\Omega_D$-logically extended membership algebra.

Homomorphisms, equivalences, and congruences of $\Omega_D$-algebras are homomorphisms, equivalences, and congruences, respectively, of membership algebras.

Given a dynamic $K$-kinded signature $\Omega_D$, define the set of *$CTL^*$ formulas over $\Omega_D$* as being the set of all $CTL^*$ formulas over the set of atomic propositions of $\Omega_D$. These formulas will be interpreted over the *multi-valued Kripke structures associated to $\Omega_D$-algebras*. More precisely, given an $\Omega_D$-algebra $\mathcal{A} = (A, \Sigma^A, \Pi_A)$, *the multi-valued Kripke structure associated to $\mathcal{A}$*, denoted $M(\mathcal{A})$, is the triple $(Q, \rightarrow_A, L_A)$, where:

- $Q = A_{state}$;
- $\rightarrow_A (q, q')$ is the value of the predicate $\rightarrow$ in $\mathcal{A}$, over $(q, q')$;
- $L_A(q, p)$ is the value of the predicate $p$ in $\mathcal{A}$, over $q$.

Given a dynamic $\Omega_D$-algebra $\mathcal{A}$ and a $CTL^*$ formula $\phi$ over $\Omega_D$, define $\mathcal{I}_{\mathcal{A}}(\phi, \pi) = [\phi]_{\pi}^{M(\mathcal{A})}$ and $\mathcal{I}_{\mathcal{A}}(\phi, q) = [\phi]_{q}^{M(\mathcal{A})}$, for any path $\pi$ and state $q$ of $M(\mathcal{A})$.

Specifications of dynamic algebras are simply membership specifications where sentences for atomic propositions and for the transition predicate are separately given.

**Definition 4.3** A *dynamic specification* is a tuple

$$DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow}),$$

where:

1. $(\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ is a dynamic $K$-kinded signature;
2. $X$ is a disjoint family of variables, disjoint of $\Sigma$ too;
3. $E$ is a set of sentences over $X$ and the memberhip signature $\Omega = (\Sigma, \pi)$ that does not contain the operator $(\_, \_)$ and the sorts $s_{p,b}$;

4. $E_{AP}$ is a set of sentences of the form:

   (a) $t : s_{p,b}$ `if` $\mathcal{C}$;

   (b) $x : s_{p,b_1}$ `if` $x : s_{p,b_2}$,

   where $t$ is a term of kind `state` over $\Omega = (\Sigma, \pi)$ and $X$, $x$ is a variable of kind `state`, $p$ is an atomic proposition, $b, b_1, b_2 \in B - \{0'\}$ with $b_1 \prec' b_2$, and $\mathcal{C}$ is a set of atomic formulas over $\Omega = (\Sigma, \pi)$ and $X$. Moreover, $E_{AP}$ contains exactly one sentence of type (4b) for every $p$ and $b_1 \leq' b_2$;

5. $E_{\rightarrow}$ is a set of sentences of the form:

   (a) $(t, t') : s_{\rightarrow,b}$ `if` $\mathcal{C}$;

   (b) $x : s_{\rightarrow,b_1}$ `if` $x : s_{\rightarrow,b_2}$,

   where $(t, t')$ is a term of kind `step`, $x$ is a variable of kind `step`, and $\mathcal{C}$ is a set of atomic formulas, all of them over $\Omega = (\Sigma, \pi)$ and $X$. Moreover, $E_{\rightarrow}$ contains exactly one sentence of type (5b) for every $b_1 \leq' b_2$.

The sentences defining the atomic propositions and the transition predicate should satisfy the following *consistency requirement*: for any predicate $p \in \Sigma_L$ and $b_1, b_2 \in B - \{0'\}$ such that there exist $x \in B$ with $x \prec' b_1$ and $x \prec' b_2$, there exist no term $t$ with $E \cup E_{AP} \cup E_{\rightarrow} \vdash t : s_{p,b_1}$ and $E \cup E_{AP} \cup E_{\rightarrow} \vdash t : s_{p,b_2}$.

**Definition 4.4** Let $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ be a dynamic specification. The *initial semantics* of $DSp$ is the class $\mathcal{M}(DSp)$ of all $\Omega_D$-algebras isomorphic to $\mathcal{T}_{\Omega_D, E'}$, where $E' = E \cup E_{AP} \cup E_{\rightarrow}$. $\mathcal{M}(DSp)$ is called an *abstract dynamic data type*.

Since dynamic specifications are membership specifications, we have:

**Proposition 4.1** Let $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ be a dynamic specification. Then, $\mathcal{T}_{\Omega_D, E'}$, where $E' = E \cup E_{AP} \cup E_{\rightarrow}$, is an initial algebra in the initial semantics of $DSp$.

**Example 4.1** The system in Figure 4.1 consists of three processes, $P_1$, $P_2$, and $P_3$, which share a memory cell $z$ and run in parallel. $P_1$ and $P_2$ may only write $z$ and their access to $z$ is mutually excluded (by a bakery-like algorithm [111]). $P_1$ increments $z$ by 2 and $P_2$ increments $z$ by 3. $P_3$ may only read $z$ and take an action according to a given value $m$ which is a multiple of 2 and 3. It is known that all readings of values less than or equal to $m$ of a memory cell $z$ are correct. If $z < m$, then $P_3$ does not take any action, and

if $z = m$, then $P_3$ definitely sets $b$ to 1. However, if $z > m$, then $P_3$ may perform an erroneous reading. In this case, it *may or may not set* $b$ to 1. To model this situation, we use the truth value $\perp$ from Kleene's three valued interpretation. The statement "*may* <statement> *end_may*" in Figure 4.1 specifies this last case.

A counter $t$ is incremented every time $P_3$ reads correctly the memory ("$t$ :=?" in Figure 4.1 means that $t$ is initially uninitialized). These three processes may have different writing/reading speeds and, therefore, the number of readings performed by $P_3$ may be arbitrarily large in comparison with the number of writings performed by $P_1$ and $P_2$. As the process $P_3$ does not take any action for $z < m$, we do not know the exact number of readings. For this reason, when $z$ becomes greater or equal to $m$, $P_3$ generates a random number $t$ (which simulates $t$ possible readings for $z < m$) and then increments it accordingly.

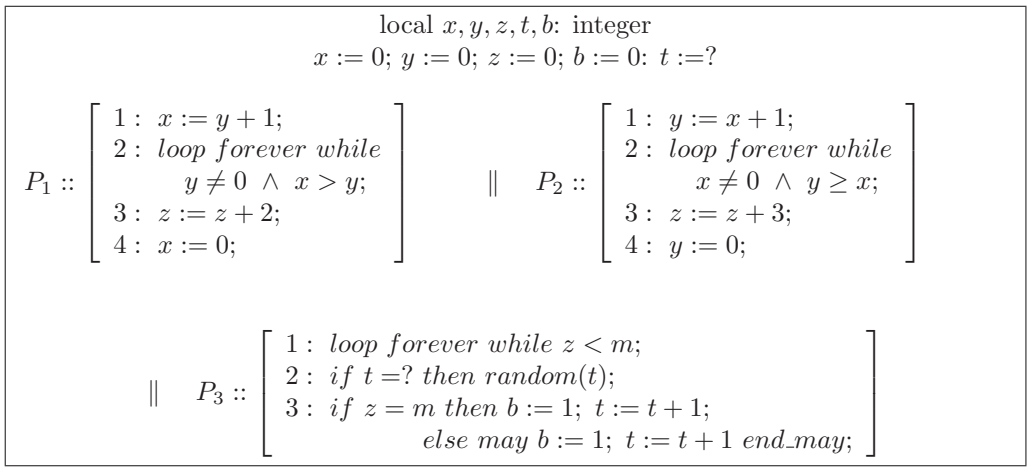


Figure 4.1: A malfunction system

We model this system by a 3-valued Kripke structure whose states are 9-dimensional vectors $(x, x', y, y', a, z, z', b, t)$ with the following meaning:

- $x$ and $y$ are the corresponding numbers of $P_1$ and $P_2$, respectively;
- $x'$ gives the local state of $P_1$: $x'$ is 0 if $x = 0$, it is 1 if $x > 0$ and $P_1$ is not writing $z$, and it is 2 if $x > 0$ and $P_1$ is writing $z$. $y'$ has a similar meaning;
- $a$ is a flag whose value is 0 when $x = y$, 1 when $x > y$, and 2 when $x < y$;

- $z$ denotes the memory content;
- $z'$ is a flag whose value is 0 when $z < m$, 1 when $z = m$, and 2 when $z > m$;
- $b$ denotes the flag that it is set to 1 by $P_3$ when $z = m$;
- $t$ stands for how many times $P_3$ reads the memory $z$.

A fragment of the Kripke structure associated to this system for $m = 6$ is represented in Figure 4.2 ($R(q, q') = 1$ is represented by an arrow from $q$ to $q'$, while $R(q, q') = \perp$ is represented by an arrow from $q$ to $q'$ labeled by $\perp$).

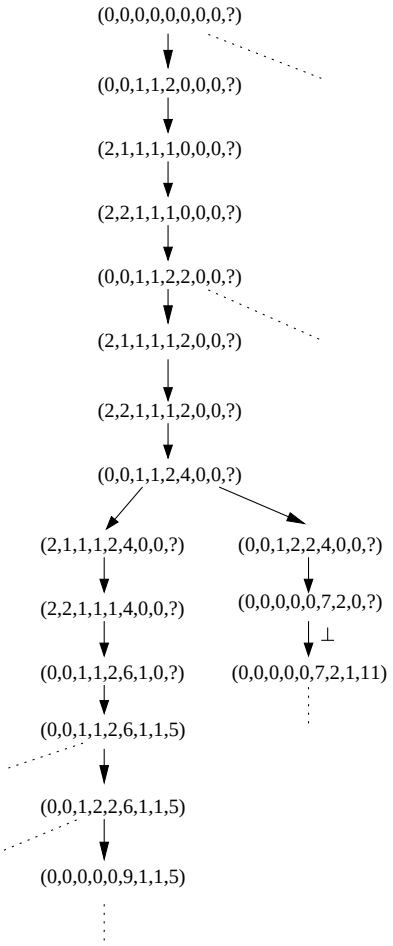


Figure 4.2: A fragment of the Kripke structure in Example 4.1

We will discuss the truth value of two LTL formulas under Kleene's three-valued interpretation:

- "$\mathbf{F}(t \geq 0) \Rightarrow \mathbf{F}(b = 1)$". This formula is true on any path which starts in $(0, 0, 0, 0, 0, 0, 0, 0, ?)$ and passes through $(0, 0, 0, 0, 0, 9, 1, 1, 5)$, and it has the truth value $\perp$ on any path which starts in $(0, 0, 0, 0, 0, 0, 0, 0, ?)$ and passes through $(0, 0, 0, 0, 0, 7, 2, 1, 11)$;

- "$\mathbf{F}(t \geq 0) \Rightarrow \mathbf{F}(t = 1)$". This formula interprets to 0 on any path which starts in $(0, 0, 0, 0, 0, 0, 0, 0, ?)$ and passes through the state $(0, 0, 0, 0, 0, 7, 2, 1, 11)$.

A dynamic specification of the program above is given below. The truth algebra is $\mathcal{B}_3 = (B, \wedge, \vee, \neg)$, where $B = \{0, \perp, 1\}$ is the set of truth values from Kleene's three-valued interpretation (as usual $0 \leq \perp \leq 1$). Moreover, we take $\leq' = \leq$. We use two kinds, `nat?`, for the set $\mathbf{N} \cup \{?\}$ ("?" stands for "uninitialized counter"), and `state`, for the set of global states. The sort `nat` of kind `nat?` will be used to represent the set of natural numbers.

Consider the following atomic propositions:

- $p$, which is 1 in states with $t \geq 0$ and it is $\perp$, otherwise;

- $q$, which is 1 in states with $b = 1$ and it is 0, otherwise;

- $r$, which is 1 in states with $t = 1$, $\perp$ in states with $t =?$, and 0, otherwise.

The sorts used to specify them are $s_{p,\perp}$, $s_{p,1}$, $s_{q,\perp}$, $s_{q,1}$, $s_{r,\perp}$, and $s_{r,1}$. The properties above can now be restated as $\mathbf{F}p \Rightarrow \mathbf{F}q$ and $\mathbf{F}p \Rightarrow \mathbf{F}r$.

In order to draw up a dynamic specification for this system we will make use of the following notations:

- we write 0 instead of $Zero$, 1 instead of $Succ(Zero)$ etc.;

- we use $t \xrightarrow{1} t'$ ($t \xrightarrow{\perp} t'$) instead of $(t, t') : s_{\rightarrow,1}$ ($(t, t') : s_{\rightarrow,\perp}$);

- we use $t \xrightarrow{1/\perp} t'$ `if` $\mathcal{C}$ whenever both $t \xrightarrow{1} t'$ `if` $\mathcal{C}$ and $t \xrightarrow{\perp} t'$ `if` $\mathcal{C}$ are elements of the specification.

**DSpec:** A Malfunction System
**kinds:** `nat?`
`state`
**sorts:** `nat` of kind `nat?`
$s_{p,\perp}, s_{p,1}, s_{q,\perp}, s_{q,1}, s_{r,\perp}, s_{r,1}$ of kind `state`
**opns:** $Succ$ : `nat?` $\rightarrow$ `nat?`
$(\_, \_, \_, \_, \_, \_, \_, \_, \_)$ : $(\texttt{nat?})^9 \rightarrow$ `state`

**vars:** $x, x', y, y', z, z', a, b, t$ : `nat?`
$u$ : `state`
$v$ : `step`

**E**: sentences for `nat`

**E**$_{AP}$**:**
$(x, x', y, y', a, z, z', b, t) : s_{p,\perp}$
$(x, x', y, y', a, z, z', 1, t) : s_{q,\perp}$
$(x, x', y, y', a, z, z', b, ?) : s_{r,\perp}$
$(x, x', y, y', a, z, z', b, t) : s_{r,\perp}$ `if` $t = 1$
$(x, x', y, y', a, z, z', b, t) : s_{p,1}$ `if` $t$ : `nat`
$(x, x', y, y', a, z, z', 1, t) : s_{q,1}$
$(x, x', y, y', a, z, z', b, t) : s_{r,1}$ `if` $t = 1$
$u : s_{p,\perp}$ `if` $u : s_{p,1}$
$u : s_{q,\perp}$ `if` $u : s_{q,1}$
$u : s_{r,\perp}$ `if` $u : s_{r,1}$

**E**$_{\rightarrow}$:
$(0, 0, 0, 0, 0, 0, 0, 0, ?) \xrightarrow{1/\perp} (1, 1, 1, 1, 0, 0, 0, 0, ?)$
$(0, 0, y, y', a, z, z', b, t) \xrightarrow{1/\perp} (Succ(y), 1, y, y', 1, z, z', b, t)$
$(x, x', 0, 0, a, z, z', b, t) \xrightarrow{1/\perp} (x, x', Succ(x), 1, 2, z, z', b, t)$
$(1, 1, 1, 1, 0, 0, 0, 0, ?) \xrightarrow{1/\perp} (1, 2, 1, 1, 0, 0, 0, 0, ?)$
$(x, 1, y, 1, 1, z, z', b, t) \xrightarrow{1/\perp} (x, 2, y, 1, 1, z, z', b, t)$
$(x, 1, y, 1, 2, z, z', b, t) \xrightarrow{1/\perp} (x, 1, y, 2, 2, z, z', b, t)$
$(0, 0, y, 1, 2, z, z', b, t) \xrightarrow{1/\perp} (0, 0, y, 2, 2, z, z', b, t)$
$(x, 1, 0, 0, 1, z, z', b, t) \xrightarrow{1/\perp} (x, 2, 0, 0, 1, z, z', b, t)$
$(x, 2, y, y', a, z, 0, 0, ?) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 0, 0, ?)$
`if` $Succ(Succ(z)) = 2,\ y$ : `nat`
$\cdots$
$(x, 2, y, y', a, z, 0, 0, ?) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 0, 0, ?)$
`if` $Succ(Succ(z)) = m - 1,\ y$ : `nat`
$(x, 2, y, y', a, z, 0, b, t) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 1, b, t)$
`if` $Succ(Succ(z)) = m,\ y$ : `nat`
$(x, 2, y, y', a, z, 0, b, t) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 2, b, t)$
`if` $Succ(z) = m,\ y$ : `nat`
$(x, 2, y, y', a, z, 1, b, t) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 2, b, t)$
`if` $y$ : `nat`
$(x, 2, y, y', a, z, 2, b, t) \xrightarrow{1/\perp} (0, 0, y, y', 2, Succ(Succ(z)), 2, b, t)$
`if` $y$ : `nat`
$(x, 2, 0, 0, 1, z, 0, 0, ?) \xrightarrow{1/\perp} (0, 0, 0, 0, 0, Succ(Succ(z)), 0, 0, ?)$

$\texttt{if } Succ(Succ(z)) = 2$

$\cdots$

$(x, 2, 0, 0, 1, z, 0, 0, ?) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(z)), 0, 0, ?)$

$\texttt{if } Succ(Succ(z)) = m - 1$

$(x, 2, 0, 0, 1, z, 0, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(z)), 1, b, t)$

$\texttt{if } Succ(Succ(z)) = m$

$(x, 2, 0, 0, 1, z, 0, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(z)), 2, b, t)$

$\texttt{if } Succ(z) = m$

$(x, 2, 0, 0, 1, z, 1, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(z)), 2, b, t)$

$(x, 2, 0, 0, 1, z, 2, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(z)), 2, b, t)$

$(x, x', y, 2, a, z, 0, 0, ?) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 0, 0, ?)$

$\texttt{if } Succ(Succ(Succ(z))) = 3,\ x : \texttt{nat}$

$\cdots$

$(x, x', y, 2, a, z, 0, 0, ?) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 0, 0, ?)$

$\texttt{if } Succ(Succ(Succ(z))) = m - 1,\ x : \texttt{nat}$

$(x, x', y, 2, a, z, 0, b, t) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 1, b, t)$

$\texttt{if } Succ(Succ(Succ(z))) = m,\ x : \texttt{nat}$

$(x, x', y, 2, a, z, 0, b, t) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 2, b, t)$

$\texttt{if } Succ(Succ(z)) = m,\ x : \texttt{nat}$

$(x, x', y, 2, a, z, 0, b, t) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 2, b, t)$

$\texttt{if } Succ(z) = m,\ x : \texttt{nat}$

$(x, x', y, 2, a, z, 1, b, t) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 2, b, t)$

$\texttt{if } x : \texttt{nat}$

$(x, x', y, 2, a, z, 2, b, t) \xrightarrow{1/\bot} (x, x', 0, 0, 1, Succ(Succ(Succ(z))), 2, b, t)$

$\texttt{if } x : \texttt{nat}$

$(0, 0, y, 2, 2, z, 0, 0, ?) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(Succ(z))), 0, 0, ?)$

$\texttt{if } Succ(Succ(Succ(z))) = 3$

$\cdots$

$(0, 0, y, 2, 2, z, 0, 0, ?) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(Succ(z))), 0, 0, ?)$

$\texttt{if } Succ(Succ(Succ(z))) = m - 1$

$(0, 0, y, 2, 2, z, 0, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(Succ(z))), 1, b, t)$

$\texttt{if } Succ(Succ(Succ(z))) = m$

$(0, 0, y, 2, 2, z, 0, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(Succ(z))), 2, b, t)$

$\texttt{if } Succ(Succ(z)) = m$

$(0, 0, y, 2, 2, z, 0, b, t) \xrightarrow{1/\bot} (0, 0, 0, 0, 0, Succ(Succ(Succ(z))), 2, b, t)$

$$
\begin{array}{l}
\qquad \texttt{if } Succ(z) = m \\
(0,0,y,2,2,z,1,b,t) \xrightarrow{1/\bot} (0,0,0,0,0,Succ(Succ(Succ(z))),2,b,t) \\
(0,0,y,2,2,z,2,b,t) \xrightarrow{1/\bot} (0,0,0,0,0,Succ(Succ(Succ(z))),2,b,t) \\
(x,x',y,y',a,z,1,0,?) \xrightarrow{1/\bot} (x,x',y,y',a,z,1,1,t) \\
(x,x',y,y',a,z,2,0,?) \xrightarrow{\bot} (x,x',y,y',a,z,2,1,t) \\
(x,x',y,y',a,z,2,1,t) \xrightarrow{\bot} (x,x',y,y',a,z,2,1,Succ(t)) \\
v : s_{\rightarrow,\bot} \texttt{ if } v : s_{\rightarrow,1}
\end{array}
$$

## 4.2 Abstractions and Preservation Results

In the previous section, a multi-valued Kripke structure $M(\mathcal{A})$ was associated to an $\Omega_D$-algebra $\mathcal{A}$. If we perform an abstraction on $\mathcal{A}$ driven by a congruence $\rho$ and two interpretation policies $\alpha_R$ and $\alpha_L$, then the Kripke structure associated to the abstraction is obtained by applying $\rho$ to $M(\mathcal{A})$ and by reinterpreting the transition predicate according to $\alpha_R$ and the atomic propositions according to $\alpha_L$. We can use the relationships between the two Kripke structures established in Section 2.3 in order to obtain preservation results for the corresponding dynamic data types.

**Definition 4.5** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_D$-algebra, $\rho$ a congruence on $\mathcal{A}$, and $\alpha_R$, $\alpha_L$ two interpretation policies over $\mathcal{B}$. The $(\alpha_R, \alpha_L)$-*abstraction of* $\mathcal{A}$ *by* $\rho$ is an $\Omega_D$-algebra $\mathcal{B} = (A/\rho, \Sigma^{A/\rho}, \Pi')$ such that:

- $\Pi'(s) = \{[a]_{\rho_{\pi(s)}} \mid [a]_{\rho_{\pi(s)}} \cap \Pi_A(s) \neq \emptyset\}$, for any sort $s$ not representing some predicate;
- $[a]_\rho \in \Pi'(s_{\rightarrow,b})$, for some $a \in A_{\texttt{step}}$ and $b \in B$, if there exists $b' \in B$ such that $b \leq' b'$ and $b'$ is the value of $\rightarrow$ over $[a]_\rho$ according to $\alpha_R$;
- $[a]_\rho \in \Pi'(s_{p,b})$, for some $p \in \Sigma_{L,\texttt{state}}$, $a \in A_{\texttt{state}}$ and $b \in B$, if there exists $b' \in B$ such that $b \leq' b'$ and $b'$ is the value of $p$ over $[a]_\rho$ according to $\alpha_L$.

Clearly, abstractions of dynamic data types correspond to abstractions of multi-valued Kripke structures.

**Theorem 4.1** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature and $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_D$-algebra. If $\mathcal{B}$ is an $(\alpha_R, \alpha_L)$-abstraction of $\mathcal{A}$ by $\rho$, then $M(\mathcal{B})$ is an $(\alpha_R, \alpha_L)$-abstraction of $M(\mathcal{A})$ by $\rho$.

**Proof** Directly from Definition 2.21 and 4.5. □

Theorem 4.1 permits us to extend all the preservation results that hold for abstractions of multi-valued Kripke structures to abstractions of dynamic data types.

**Theorem 4.2** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D \subseteq B$, $\mathcal{B}$ an $(\alpha_R, \alpha_L)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL^*_+$ formula over $\Sigma_{L,\mathtt{state}}$. If

1. $(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset)$, for any $b \in B - D$;
2. $(\alpha_L(d) = \exists^S \;\Rightarrow\; S \subseteq D) \;\wedge\; (\alpha_L(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq D)$, for any $d \in D$;
3. $D$ is closed under $\leq$ and glb, and backward closed under lub,

then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \in D \;\;\Rightarrow\;\; \mathcal{I}_{\mathcal{A}}(\phi, q) \in D)$$

and

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}), D))(\mathcal{I}_{\mathcal{B}}(\psi, \pi_2) \in D \;\Rightarrow\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi, \pi_1) \in D)).$$

**Proof** Directly from Theorem 2.15 and Remark 2.4. □

**Theorem 4.3** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D \subseteq B$, $b \in B$, $\mathcal{B}$ an $(\alpha_R, \alpha_L)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL^*_+$ formula over $\Sigma_{L,\mathtt{state}}$. If

1. $(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset)$, for any $b \in B - D$;
2. $(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow)$, for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \geq b$;
3. for any subset $B'$ of $B$, $\vee B' \geq b$ implies $b' \geq b$ for some $b' \in B'$,

then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \geq b \;\;\Rightarrow\;\; \mathcal{I}_{\mathcal{A}}(\phi, q) \geq b)$$

and

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}), D))(\mathcal{I}_{\mathcal{B}}(\psi, \pi_2) \geq b \;\;\Rightarrow\;\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi, \pi_1) \geq b)).$$

**Proof** Directly from Theorem 2.16 and Remark 2.5. □

The preservation results for dynamic data types over Kleene's three-valued interpretation ($\mathcal{B}_3$ denotes the corresponding truth algebra) from [46] are obtained as particular cases. The following interpretation policies are considered in [46]:

- $\alpha_1(0) = \forall$, $\alpha_1(\bot) = \exists_a^{\{0,\bot,1\}}$ and $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \exists^{\{0,\bot,1\}}$, $\alpha_2(\bot) = \exists^{\{\bot,1\}}$ and $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\bot) = \exists^{\{0,\bot\}}$ and $\alpha_3(1) = \exists^{\{0,\bot,1\}}$.

**Corollary 4.1** Let $\Omega_D = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D = \{\bot, 1\}$, $\mathcal{B}$ an $(\alpha_3, \alpha_2)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ an mv-$LTL_+$ formula over $\Sigma_{L,\mathtt{state}}$. Then,

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \geq \bot \quad \Rightarrow \quad \mathcal{I}_{\mathcal{A}}(\phi, q) \geq \bot).$$

**Proof** Directly from Theorem 4.3. The second condition must be satisfied only for $\alpha = \alpha_L$ because $D = \{\bot, 1\}$. □

**Corollary 4.2** Let $\Omega_D = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D = \{\bot, 1\}$, $\mathcal{A}$ an $\Omega_D$-algebra, and $\phi$ an mv-$LTL_+$ formula over $\Sigma_{L,\mathtt{state}}$. We have the following:

- if $\mathcal{B}$ is an $(\alpha_3, \alpha_2)$- or an $(\alpha_3, \alpha_1)$-abstraction of $\mathcal{A}$ by a congruence $\rho$ and $M(\mathcal{A})$ does not contain any arcs labeled by $\bot$, then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) = 1 \quad \Rightarrow \quad \mathcal{I}_{\mathcal{A}}(\phi, q) = 1);$$

- if $\mathcal{B}$ is an $(\alpha_1, \alpha_2)$- or an $(\alpha_1, \alpha_1)$-abstraction of $\mathcal{A}$ by a congruence $\rho$, then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) = 1 \quad \Rightarrow \quad \mathcal{I}_{\mathcal{A}}(\phi, q) = 1);$$

**Proof** The first part follows from Corollary 2.10 and the second from Theorem 4.3. □

**Theorem 4.4** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D \subseteq B$, $b \in B$, $\mathcal{B}$ an $(\alpha_R, \alpha_L)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\forall CTL_+^*$ formula over $\Sigma_{L,\mathtt{state}}$. If

1. $(\alpha_R(b) = \exists^S \ \Rightarrow \ S \subseteq D) \ \wedge \ (\alpha_R(b) = \exists_a^S \ \Rightarrow \ S \cup b\downarrow \cup b\uparrow \subseteq D)$, for any $b \in D$;

2. $(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \wedge (\alpha(d) = \exists_a^S \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow)$, for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \leq b$;

3. for any subset $B'$ of $B$, $\wedge B' \leq b$ implies $b' \leq b$ for some $b' \in B'$,

then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \leq b \;\Rightarrow\; \mathcal{I}_{\mathcal{A}}(\phi, q) \leq b)$$

and

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}), D))(\mathcal{I}_{\mathcal{B}}(\psi, \pi_2) \leq b \;\Rightarrow\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi, \pi_1) \leq b)).$$

**Proof** Directly from Theorem 2.18 and Remark 2.6. □

In the following, we give another preservation result from [46] which can be obtained as a particular case.

**Corollary 4.3** Let $\Omega_D = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D = \{\bot, 1\}$, $\mathcal{A}$ an $\Omega_D$-algebra, and $\phi$ an mv-$LTL_+$ formula over $\Sigma_{L,\mathtt{state}}$. We have the following:

- if $\mathcal{B}$ is an $(\alpha_2, \alpha_3)$- or an $(\alpha_2, \alpha_1)$-abstraction of $\mathcal{A}$ by a congruence $\rho$ then
$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) = 0 \;\Rightarrow\; \mathcal{I}_{\mathcal{A}}(\phi, q) = 0);$$
- if $\mathcal{B}$ is an $(\alpha_1, \alpha_3)$- or an $(\alpha_1, \alpha_1)$-abstraction of $\mathcal{A}$ by a congruence $\rho$ and $\phi \in LTL_+^{\infty}$ then
$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) = 0 \;\Rightarrow\; \mathcal{I}_{\mathcal{A}}(\phi, q) = 0).$$

**Proof** The first part follows from Theorem 4.4 while the second from Corollary 2.12. □

**Theorem 4.5** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D \subseteq B$, $b \in B$, $\mathcal{B}$ an $(\alpha_R, \alpha_L)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\exists CTL_+^*$ formula over $\Sigma_{L,\mathtt{state}}$. If

1. $(\alpha_R(b) = \exists^S \;\Rightarrow\; S \subseteq D) \wedge (\alpha_R(b) = \exists_a^S \;\Rightarrow\; S \cup b\downarrow \cup b\uparrow \subseteq D)$, for any $b \in D$;

2. $(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\uparrow) \wedge (\alpha(d) = \exists_a^S \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\uparrow)$, for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \geq b$;

3. for any subset $B'$ of $B$, $\vee B' \geq b$ implies $b' \geq b$ for some $b' \in B'$,

then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \geq b \;\; \Rightarrow \;\; \mathcal{I}_{\mathcal{A}}(\phi, q) \geq b)$$

and

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}), D))(\mathcal{I}_{\mathcal{B}}(\psi, \pi_2) \geq b \;\; \Rightarrow \;\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi, \pi_1) \geq b)).$$

**Proof** Directly from Theorem 2.19 and Remark 2.7. □

**Theorem 4.6** Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $D \subseteq B$, $b \in B$, $\mathcal{B}$ an $(\alpha_R, \alpha_L)$-abstraction of an $\Omega_D$-algebra $\mathcal{A}$ by a congruence $\rho$, and $\phi$ and $\psi$ be a state and, respectively, a path mv-$\exists CTL^*_+$ formula over $\Sigma_{L,\mathtt{state}}$. If

1. $(\alpha_R(b) = \exists^S \;\Rightarrow\; S \cap D = \emptyset) \;\wedge\; (\alpha_R(b) = \exists^S_a \;\Rightarrow\; S \cap D = b\downarrow \cap D = b\uparrow \cap D = \emptyset)$, for any $b \in B - D$;
2. $(\alpha(d) = \exists^S \;\Rightarrow\; S \subseteq b\downarrow) \;\wedge\; (\alpha(d) = \exists^S_a \;\Rightarrow\; S \cup d\downarrow \cup d\uparrow \subseteq b\downarrow)$, for any $\alpha \in \{\alpha_R, \alpha_L\}$ and $d \leq b$;
3. for any subset $B'$ of $B$, $\wedge B' \leq b$ implies $b' \leq b$ for some $b' \in B'$,

then

$$(\forall q \in A_{\mathtt{state}})(\mathcal{I}_{\mathcal{B}}(\phi, [q]) \leq b \;\; \Rightarrow \;\; \mathcal{I}_{\mathcal{A}}(\phi, q) \leq b)$$

and

$$(\forall \pi_2 \in \Pi(M(\mathcal{B}), D))(\mathcal{I}_{\mathcal{B}}(\psi, \pi_2) \leq b \;\; \Rightarrow \;\; (\forall \pi_1 \in C_{M(\mathcal{A})}(\pi_2))(\mathcal{I}_{\mathcal{A}}(\psi, \pi_1) \leq b)).$$

**Proof** Directly from Theorem 2.20 and Remark 2.8. □

**Example 4.2** We illustrate the use of abstraction on the system in Example 4.1. Assume that this system is modeled by a dynamic algebra $\mathcal{A}$. We consider $\rho$ a congruence on $\mathbf{A}$ such that:

$$(x_1, x_1', y_1, y_1', a_1, z_1, z_1', b_1, t_1) \; \rho_{\mathtt{state}} \; (x_2, x_2', y_2, y_2', a_2, z_2, z_2', b_2, t_2)$$

if and only if

$$z_1' = z_2' \;\wedge\; b_1 = b_2 \;\wedge\; (t_1 = t_2 =? \;\vee\; t_1 = t_2 = 0 \;\vee\; t_1 = t_2 = 1 \;\vee\; t_1, t_2 > 1).$$

Let $\mathcal{B}$ be the $(\alpha_3, \alpha_2)$-abstraction of $\mathcal{A}$ induced by $\rho$, where

- $\alpha_2(0) = \exists^{\{0,\perp,1\}}$, $\alpha_2(\perp) = \exists^{\{\perp,1\}}$ and $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\perp) = \exists^{\{0,\perp\}}$ and $\alpha_3(1) = \exists^{\{0,\perp,1\}}$.

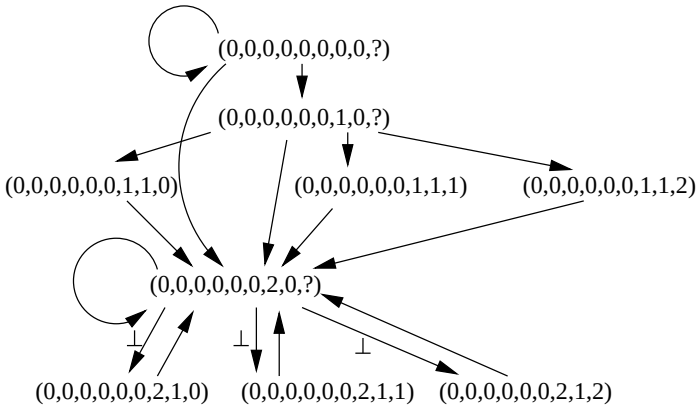


Figure 4.3: The Kripke structure for the abstraction in Example 4.2

Notice that $M(\mathcal{B})$ contains only 24 states and hence it can be easily model checked (Figure 4.2 depicts only the reachable states). For example, one can easily see that the truth value of the formula $\mathbf{F}p \Rightarrow \mathbf{F}q$ from Example 4.1 evaluates to $\perp$ in the abstract system and, therefore, it evaluates to 1 or $\perp$ in the original system $\mathcal{A}$ by Corollary 4.1.

Now, we show how to compute multi-valued abstractions of dynamic data types from 2-valued abstractions of dynamic data types. The $S'$-congruences from Definition 3.4 can be applied as well on $\Omega_D$-algebras which are particular cases of membership algebras. The quotients of dynamic algebras by $S'$-congruences can be viewed as 2-valued abstractions because the redefinitions of the sorts in the quotient are done only by under-approximation and over-approximation.

Let $\Omega_D = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic signature, $\mathcal{A}$ an $\Omega_D$-algebra, and $\rho$ a congruence on $\mathcal{A}$ such that $\leq'$ is appropriate for $\rho$. Moreover, let $S'_{AP}$ be a set of sorts representing atomic propositions such that if $s_{p,b} \in S'$, for some $p \in \Sigma_{L,\mathtt{state}}$, then

$$(\forall p \in \Sigma_{L,\mathtt{state}})(s_{p,b} \in S'),$$

and $S'_{\rightarrow}$ a set of sorts from the set $\{s_{\rightarrow,b} | b \in B\}$. We require that $S'_{AP}$ and $S'_{\rightarrow}$ are appropriate for $\rho$, which is defined similarly to the appropriateness of $\mathcal{P}$ for $\rho$ from Definition 2.5. We will denote by $B_{AP,S'}$ the set of truth values $b$ for which the sorts $s_{p,b}$ with $p \in \Sigma_{L,\mathtt{state}}$ are in $S'_{AP}$ and by $B_{\rightarrow,S'}$ the set of truth values $b$ for which the sort $s_{\rightarrow,b}$ is in $S'$.

We can prove that the quotient of $\mathcal{A}$ by the $S'$-congruence $(\rho, S')$, where $S' = S'_{AP} \cup S'_{\rightarrow}$, equals the $(\alpha_R, \alpha_L)$-abstraction of $\mathcal{A}$ by $\rho$, where $\alpha_R$ and $\alpha_L$ are obtained using the eight rules from Theorem 2.8 in which we replace $B_{\mathcal{P}}$ with $B_{\rightarrow,S'}$ and $B_{AP,S'}$, respectively.

**Example 4.3** The $(\alpha_2, \alpha_1)$-abstraction of $\mathcal{A}$ by $\rho$ equals the quotient of $\mathcal{A}$ by $(\rho, S')$, where

$$S' = \{s_{p,\perp}, s_{p,1}, s_{q,\perp}, s_{q,1}, s_{r,\perp}, s_{r,1}\}.$$

Naturally, abstractions can be applied to abstract dynamic data types $\mathcal{M}(DSp)$, where $DSp$ is a dynamic specification, by means of a representative of them, and $T_{\Omega_D, E \cup E_{AP} \cup E_{\rightarrow}}$ is a suitable choice. The abstractions are specified by equations and, consequently, they are *equationally specified abstractions*.

Let $DSp$ be a dynamic specification. The value of a predicate $p$ over a term $t$ is the maximum truth value $b \in B$ (with respect to $\leq'$) for which $E \cup E_{AP} \cup E_{\rightarrow} \vdash t : s_{p,b}$, or $0'$, if such a $b$ does not exist.

**Definition 4.6** Let $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ be a dynamic membership specification. An abstraction of $DSp$ is a triple $\Delta = (A, \alpha_R, \alpha_L)$, where $A$ is a set of sentences over $\Omega$ and $X$, and $\alpha_R$ and $\alpha_L$ are interpretation policies over $\mathcal{B}$.

**Definition 4.7** Let $DSp = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E, E_{AP}, E_{\rightarrow})$ be a dynamic membership specification and $\Delta = (A, \alpha_R, \alpha_L)$ an abstraction of it. The *dynamic membership specification for the abstraction of $DSp$ by $\Delta$* is $DSp_\Delta = (\mathcal{B}, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E^\Delta_{AP}, E^\Delta_{\rightarrow})$, where:

- $E \cup A \cup E^\Delta_{AP} \vdash t : s_{p,b}$ iff the value of $p$ over the set of terms $\{t' | E \cup A \vdash t = t'\}$ according to $\alpha_L$ is greater than or equal to $b$ (with respect to $\leq'$).
- $E \cup A \cup E^\Delta_{\rightarrow} \vdash t : s_{p,b}$ iff the value of $\rightarrow$ over the set of terms $\{t' | E \cup A \vdash t = t'\}$ according to $\alpha_R$ is greater than or equal to $b$ (with respect to $\leq'$).

Now, from the definition of $DSp_\Delta$ we can derive the following result which states that the initial semantics of $DSp_\Delta$ contains dynamic algebras that are $(\alpha_R, \alpha_L)$-abstractions of the dynamic algebras from the initial semantics of $DSp$.

**Proposition 4.2** Let $DSp$, $\Delta$ and $DSp_\Delta$ be as above. Let $\rho$ be the congruence on $\mathcal{T}_{\Omega_D, E'}$, where $E' = E \cup E_{AP} \cup E_{\rightarrow}$, defined by:

$$[t]_{=_{E',k}} \ \rho_k \ [t']_{=_{E',k}} \Leftrightarrow E'' \vdash t = t', \text{ for any } t, t' \in (T_\Omega)_k,$$

where $E'' = E \cup A \cup E^\Delta_{AP} \cup E^\Delta_{\rightarrow}$. Then, $\mathcal{T}_{\Omega, E''}$ is the $(\alpha_R, \alpha_L)$-abstraction of $\mathcal{T}_{\Omega, E'}$ by $\rho$.

**Example 4.4** We exemplify the abstraction technique above on the abstract dynamic data type induced by the specification *DSpec* from Example 4.1.

We will apply an $(\alpha_2, \alpha_1)$-abstraction $\Delta$ induced by the following set $A$ of sentences:

$$
\begin{aligned}
A\colon\quad & (x_1, x_1', y_1, y_1', a_1, z_1, z', b, ?) = (x_2, x_2', y_2, y_2', a_2, z_2, z', b, ?) \\
& (x_1, x_1', y_1, y_1', a_1, z_1, z', b, 0) = (x_2, x_2', y_2, y_2', a_2, z_2, z', b, 0) \\
& (x_1, x_1', y_1, y_1', a_1, z_1, z', b, 1) = (x_2, x_2', y_2, y_2', a_2, z_2, z', b, 1) \\
& (x_1, x_1', y_1, y_1', a_1, z_1, z', b, Succ(Succ(t))) = \\
& \qquad\qquad (x_2, x_2', y_2, y_2', a_2, z_2, z', b, Succ(Succ(Succ(t))))
\end{aligned}
$$

As we can easily see the dynamic specification for the abstraction of *DSpec* by $\Delta$ is $DSpec_\Delta = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi, X, E \cup A, E_{AP}, E_{\rightarrow})$.

# Chapter 5

# Abstraction Refinement Techniques

The main problem that arises when using abstraction techniques is to find the suitable abstraction or to refine an already existing abstraction in order to obtain a better one [76, 105, 66, 25, 87, 3, 80]. In this chapter, we prove that the abstraction techniques for data types, under Kleene's three-valued interpretation [111], can be used in a refinement procedure. Moreover, we prove that the counterexample guided abstraction refinement procedure [25] works better when used with equationally specified abstractions.

## 5.1 Abstraction Refinement for Data Types

We present a result adapted from [111] which proves that we can define an abstraction of a data type[1] starting from another abstraction of the same data type. This implies in fact the construction of a quotient algebra starting from a quotient algebra and a congruence. Such a two-step reduction should be also definable by a one-step reduction.

Let $\mathcal{B}_3$ be the truth algebra corresponding to Kleene's three-valued logic and $\leq'=\leq$. If $\Omega_L$ is a logically extended signature, $\mathcal{A}$ an $\Omega_L$-algebra and $\theta$, $\rho$ two congruences on $\mathcal{A}$, we say that $\theta$ is *finer* than $\rho$ if $\theta \subseteq \rho$.

Figure 5.1 gives a pictorial view of the property "finer than" between congruences. One can easily prove (see also [89]) that the binary relation $\rho/\theta$ given by

$$(\rho/\theta)_k = \{([a]_{\theta_k}, [b]_{\theta_k}) | (a, b) \in \rho_k\},$$

[1] We refer to abstractions preserving formulas in a first order logic under Kleene's three-valued logic.

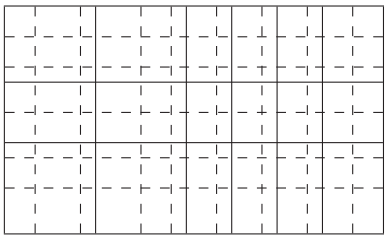


Figure 5.1: $\theta$ is finer than $\rho$

for all kinds $k$, is a congruence on $\mathcal{A}/\theta$ whenever $\theta$ and $\rho$ are congruences on $\mathcal{A}$ such that $\theta \subseteq \rho$.

The following safe interpretation policies are considered in [111]:

- $\alpha_1(0) = \forall$, $\alpha_1(\bot) = \exists_a^{\{0,\bot,1\}}$ and $\alpha_1(1) = \forall$;
- $\alpha_2(0) = \exists^{\{0,\bot,1\}}$, $\alpha_2(\bot) = \exists^{\{\bot,1\}}$ and $\alpha_2(1) = \forall$;
- $\alpha_3(0) = \forall$, $\alpha_3(\bot) = \exists^{\{0,\bot,1\}}$ and $\alpha_3(1) = \exists^{\{0,1\}}$.

**Theorem 5.1** Let $\Omega_L = (\mathcal{B}_3, \leq', \Sigma, \Sigma_L, \pi)$ be a logically extended signature, $\mathcal{A} = (A, \Sigma^A, \Pi_A)$ an $\Omega_L$-algebra, $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$, $\mathcal{B}_1$ an $\alpha$-abstraction of $\mathcal{A}$ by a congruence $\theta$, and $\mathcal{B}_2$ an $\alpha$-abstraction of $\mathcal{A}$ by a congruence $\rho$. If $\theta$ is finer than $\rho$ then there exists $\mathcal{B}$ an $\alpha$-abstraction of $\mathcal{B}_1$ by a congruence $\delta$ such that $\mathcal{B} \cong \mathcal{B}_2$ and $\mathcal{B}_2 \cong \mathcal{B}$.

**Proof** Let $\delta$ be a congruence on $\mathcal{B}_1$ such that $\delta = \rho/\theta$ and let $h : (A/\theta)_{/\delta} \to A/\rho$ be a mapping defined by

$$h([[a]_\theta]_\delta) = [a]_\rho,$$

for all $a \in A$.

Following a classical line (for example, [89]) one can easily prove that $h$ is a bijective homomorphism between the $\Sigma$-algebras $\mathbf{A}/\rho$ and $(\mathbf{A}/\theta)_{/\delta}$. Therefore, the following properties remain to be proved:

- $h(\Pi_\mathcal{B}(s_{p,1})) = \Pi_{\mathcal{B}_2}(s_{p,1})$ and $h(\Pi_\mathcal{B}(s_{p,\bot})) = \Pi_{\mathcal{B}_2}(s_{p,\bot})$, for any $p \in \Sigma_L$.

But, these properties can be easily checked for each type of abstraction.

For example, in the case of $\alpha = \alpha_1$, let $([[a_1]_\theta]_\delta, \ldots, [[a_n]_\theta]_\delta) \in \Pi_\mathcal{B}(s_{p,1})$, for some $p$ of type $k_1 \cdots k_n$ and $a_1 \in A_{k_1}, \ldots, a_n \in A_{k_n}$. By the definition of $\mathcal{B}$, we obtain that $([b_1]_\theta, \ldots, [b_n]_\theta) \in \Pi_{\mathcal{B}_1}(s_{p,1})$, for all $[b_i]_\theta \in [[a_i]_\theta]_\delta$, $1 \leq i \leq n$, and further, by the definition of $\mathcal{B}_1$, $(a'_1, \ldots, a'_n) \in \Pi_\mathcal{A}(s_{p,1})$, for all $a'_i \in [[a_i]_\theta]_\delta$, $1 \leq i \leq n$. Consequently, $(a'_1, \ldots, a'_n) \in \Pi_\mathcal{A}(s_{p,1})$, for all $a'_i \in [a_i]_\rho$, $1 \leq i \leq n$ and $([a_1]_\rho, \ldots, [a_n]_\rho) \in \Pi_{\mathcal{B}_2}(s_{p,1})$.

The other cases follow a similar line. □

Theorem 5.1 is an extension of the second homomorphism theorem for classical universal algebras [89] to logically extended membership algebras. It shows us that, in order to pass from an abstraction by a congruence $\theta$ to an abstraction by a congruence $\rho$ one can abstract further $\mathcal{A}/\theta$ by a congruence $\rho/\theta$.

## 5.2 CEGAR is Better under Equational Abstraction

A very popular technique to discover abstractions automatically is counterexample guided abstraction refinement [25] (CEGAR, for short) which starts with an initial abstraction and then, uses counterexamples found in the verification process to refine the current abstraction. The various CEGAR techniques introduced in the literature deal mainly with predicate abstraction.

In this section, we introduce a CEGAR procedure for equational abstractions of 2-valued Kripke structures. Kripke structures are represented by rewrite theories as in [92]. However, we improve the representation of the atomic propositions, so it will not lead, by itself, to useless abstractions. Moreover, the semantics of the temporal logic considers infinite paths and maximal finite paths.

The refinement procedure adds atomic formulas in the conditions of the equations that form the abstraction. As opposed to the approach that uses predicate abstraction, where we add at least one predicate, the number of states of the current abstraction may not necessarily at least double. This will imply, as proved by a consistent example, that this new refinement procedure may build smaller abstractions.

### 5.2.1 Rewrite Theories

To represent systems we use rewrite theories [90] that contain membership equational theories in which we distinguish some of the membership sentences. We denote by $\mathcal{B}_2 = (\{0,1\}, \wedge, \vee, \neg)$ the truth algebra with only two elements and by $\leq'$ the partial order defined by $0 \leq' 1$.

**Definition 5.1** Let $\Omega_D = (\mathcal{B}_2, \leq', \Sigma, \Sigma_L, \pi)$ be a dynamic $K$-kinded signature. An $\Omega_D$ *rewrite theory* over a $K$-kinded set of variables $X$, is a tuple $\mathcal{R} = (E, E_{AP}, \mathcal{E})$, where

- $E$ is a set of sentences over $X$ and the memberhip signature $\Omega = (\Sigma, \pi)$ that does not contain the operator $(\_, \_)$ and the sorts $s_{p,b}$. They specify the data type for the set of states;

- $E_{AP}$ is a set of sentences of the form

$$t : s_p \texttt{ if } \mathcal{C},$$

where $t$ is a term of kind `state` over $\Omega$ and $X$, $p \in \Sigma_L$ is an atomic proposition, and $\mathcal{C}$ is a set of atomic formulas over $\Omega$ and $X$. They will specify the sorts that represent the atomic propositions;

- $\mathcal{E}$ is a set of conditional rewriting rules of the form:

$$t \longrightarrow t' \texttt{ if } \bigwedge_{i \in I} u_i : s_i \wedge \bigwedge_{j \in J} v_j = w_j \wedge \bigwedge_{l \in L} t_l \longrightarrow t'_l,$$

where $t, t', t_l, t'_l$, for any $l \in L$ are terms of kind `state`, $u_i$, for any $i \in I$, and $v_j, w_j$, for any $j \in J$, are terms of some kind, and $s_i$, for any $i \in I$, are sorts. They will represent the transitions of the system.

We emphasize that we do not represent the set of truth values as a kind and the atomic propositions by means of a function symbol as it was done in [92] because it can lead to useless abstractions as it was already mentioned in Remark 3.1.

Given a dynamic $K$-kinded signature $\Omega_D$, define the set of *$LTL$ formulas over $\Omega_D$* as being the set of $LTL$ formulas over the set of atomic propositions of $\Omega_D$. These formulas will be interpreted over the *Kripke structures associated to $\Omega_D$ rewrite theories*.

Given an $\Omega_L$ rewrite theory $\mathcal{R} = (E, E_{AP}, \mathcal{E})$, *the Kripke structure associated to $\mathcal{R}$*, denoted $M(\mathcal{R})$, is the triple $(Q, R, L)$, where:

- $Q = T_{\texttt{state}}$, where $\mathcal{T}_{\Omega_D, E \cup E_{AP}} = (T, \Sigma^T, \Pi_T)$ is the initial algebra in the class $Mod(E \cup E_{AP})$;

- $R(q, q')$ iff $E \cup \mathcal{E} \vdash^1 q \longrightarrow q'$, for any $q, q' \in Q$ ($\vdash^1$ is the syntactic deduction which uses only one rewriting rule from $\mathcal{E}$);

- $L(q, p) = 1$ if $q \in \Pi_T(s_p)$ and 0 otherwise, for any $q \in Q$ and $p \in \Sigma_L$.

Given an $\Omega_D$ rewrite theory $\mathcal{R}$ and an $LTL$ formula $\phi$ over $\Omega_D$, define $\mathcal{I}_{\mathcal{R}}(\phi, \pi) = [\phi]_{\pi}^{M(\mathcal{R})}$ and $\mathcal{I}_{\mathcal{R}}(\phi, q) = [\phi]_{q}^{M(\mathcal{R})}$, for any infinite or maximal finite path $\pi$ and any state $q$ of $M(\mathcal{R})$.

The way we obtain a Kripke structure from a rewrite theory has similar points with the approach from [92]. However, as we consider also finite paths

in the semantics of an $LTL$ formula, we do not need that the transition relation be made total and also, we use a different representation for the atomic propositions.

Obtaining the Kripke structure associated to some rewrite theory may be undecidable. However, we can restrict ourselves to classes of rewrite theories for which the extraction of the Kripke structure is decidable. For example, we can use the class of *executable rewrite theories* [92].

### Model Checking rewrite theories

One may argue that the construction of the Kripke structure for a rewrite theory may have a great complexity but in fact, we will consider an "on the fly" model-checking algorithm that may need to compute only a part of the entire Kripke structure. This is also the approach used in the Maude $LTL$ model-checker [42].

We will discuss how to build an "on the fly" model checking algorithm for $LTL$ when considering also maximal finite paths. Remember that such a model checking technique for $LTL$ [54] is build upon two algorithms. The first algorithm constructs the Buchi automaton (over the alphabet that consists in the set of atomic propositions) that corresponds to the $LTL$ formula, in the sense that it accepts only infinite words which are models of this formula while the second consists in a nested DFS search for a reachable accepting state reachable from itself in some Buchi automaton (this automaton will be the product between the input system and the Buchi automaton build in the first part).

We modify the first part by considering Buchi automata which, besides accepting states, have also final states. They accept infinite-length words for which the corresponding run passes infinitely many times through an accepting state or finite-length words for which the corresponding run ends in a final state.

Remember that the nodes of the graph from which we will extract the Buchi automaton corresponding to an $LTL$ formula that handles only infinite paths are labeled by three sets of formulas: `New` which are formulas that must hold at the current state and have not yet been processed, `Old` which are formulas that must hold in the current node and have already been processed and `Next` which are formulas that must hold in all states that are immediate successors of states satisfying the properties in `Old`. Initially, we have only one node which has $\texttt{New} = \phi$, where $\phi$ is the input $LTL$ formula, and $\texttt{Old} = \texttt{Next} = \emptyset$. At some step of the algorithm we process the current node and we expand the graph if $\texttt{New} \neq \emptyset$, depending on the formula read from `New` (if $\texttt{New} = \emptyset$ we have finished processing the node and we can consider it as a state

of the Buchi automaton). When $\phi$ is not of the form $\overline{\mathbf{X}}\psi$, for some formula $\psi$, we proceed as usual. Otherwise, when $\phi = \overline{\mathbf{X}}\psi$ we mark the current node as final and we also expand the graph as in the case $\phi = \mathbf{X}\psi$. The nested DFS search from the second part of the "on the fly" model checking algorithm consists in an usual DFS search for an accepting state followed by a second DFS search that must find a path back to the accepting state. We modify it by allowing that the first DFS search stops when finding a final state. The correctness of the modified algorithm can be proved in a similar manner to the correctness of the classical algorithm from [54].

### 5.2.2 A Motivating Example

Before, we formalize equational abstractions of rewrite theories and it's corresponding refinement technique, we provide a motivating example that illustrates how CEGAR under equational abstraction produces smaller models to be verified than CEGAR under predicate abstraction. This happens because refining in the context of predicate abstraction means adding at least one predicate to the abstraction function and consequently it means at least the doubling of the state space of the system. Using equational abstractions we may obtain finer refinements that do not necessarily double the state space.

Consider the following protocol adapted from [35] that controls the mutually exclusive access to two common resources of two concurrent processes, modeling the behavior of two mathematicians. They alternate phases of "thinking", "eating" and "drinking" regulated by the current values of two natural numbers $m$ and $n$: the first (second) mathematician has the right to enjoy his meal if $m$ is odd (even) or to enjoy his drink if $n$ is odd (even). We suppose that drinking and eating can not take place in the same time. After eating or drinking, each mathematician leaves the dining room and modifies the value of $m$ or $n$ in his own fashion. Also, when entering the eating (drinking) phase both mathematicians modify arbitrarily the value of $n$ ($m$). We want to prove that starting from the state in which the two mathematicians are thinking and $m = n = 16$, it will always be the case that at least one of the mathematicians is thinking.

A rewrite theory for this protocol is given below (we write 1 instead of $Succ(0)$, 2 instead of $Succ(Succ(0))$, etc). We use two kinds: `nat` for the set of natural numbers and `state` for the set of global states which are 4-dimensional vectors of natural numbers $(x, y, m, n)$ such that: $x = 0$ if the first mathematician is thinking, $x = 1$ if the first mathematician is eating and $x = 2$ if the first mathematician is drinking; $y$ represents the state of the second mathematician as $x$ does for the first one; $m$, $n$ are the numbers used to grant the access to the shared resources. Also, we use the sort `nat01` of

kind `nat` for the set $\{0, 1\}$ and the sort `nat02` of kind `nat` for the set $\{0, 2\}$. The only atomic proposition used is $p$ which is 1 in states with $x = 0$ or $y = 0$ (states in which at least one mathematician is thinking) and 0, otherwise. The sort used to specify it is $s_p$ and the property above can be restated as $\phi = \mathbf{G}\ p$. A fragment of the Kripke structure for the above system is shown in Figure 5.2.

**DRewTh:** A Protocol for the Mutually Exclusive Access to Two Resources

| | |
|---|---|
| **kinds:** | `nat` |
| | `state` |
| **sorts:** | `nat01`, `nat02` of kind `nat` |
| | $s_p$ of kind `state` |
| **opns:** | $0 : \rightarrow$ `nat` |
| | $Succ$ : `nat` $\rightarrow$ `nat` |
| | $\%$ : `nat`$^2 \rightarrow$ `nat` |
| | $(\_, \_, \_, \_)$ : `nat`$^4 \rightarrow$ `state` |
| **vars:** | $x, y, m, n, m', n'$ : `nat` |
| **E**: | sentences for `nat01`, `nat02`, $Succ$ and $\%$. |
| $\mathbf{E}_{AP}$**:** | $(x, y, m, n) : s_p$ `if` $x = 0$ |
| | $(x, y, m, n) : s_p$ `if` $y = 0$ |
| $\mathbf{E}_{\rightarrow}$: | $(0, y, m, n) \longrightarrow (1, y, m, n')$ `if` $m\%2 = 1,\ y$ : `nat01` |
| | $(1, y, m, n) \longrightarrow (0, y, Succ(3m), n)$ |
| | $(x, 0, m, n) \longrightarrow (x, 1, m, n')$ `if` $m\%2 = 0,\ x$ : `nat01` |
| | $(x, 1, m, n) \longrightarrow (x, 0, m/2, n)$ `if` $m\%2 = 0$ |
| | $(0, y, m, n) \longrightarrow (2, y, m', n)$ `if` $n\%2 = 1,\ y$ : `nat02` |
| | $(2, y, m, n) \longrightarrow (0, y, m, Succ(3n))$ |
| | $(x, 0, m, n) \longrightarrow (x, 2, m', n)$ `if` $n\%2 = 0,\ x$ : `nat02` |
| | $(x, 2, m, n) \longrightarrow (x, 0, m, n/2)$ `if` $n\%2 = 0$ |

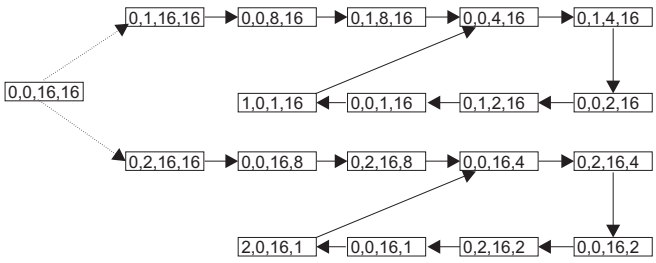


Figure 5.2: A fragment from the Kripke structure of the concrete system

Intuitively, by an equational abstraction of a rewrite theory we will mean a set of membership sentences $\Delta$. The rewrite theory for the abstract system will be obtained from the rewrite theory of the concrete one by adding

$\Delta$ to the sentences that specify the data type for the set of states and by modifying the sentences representing atomic propositions in order to obtain under-approximations. The CEGAR technique for equational abstraction will proceed as the one for predicate abstraction but, instead of adding predicates, we will add atomic formulas to some of the membership sentences of the abstraction.

In the following, we will prove $\phi$ by abstraction: first, we use CEGAR under equational abstraction and then, the classical version under predicate abstraction. The final abstractions obtained by the two procedures will show that the approach using equational abstraction is significantly better from the point of view of the number of states and transitions than the one that uses predicate abstraction.

### The approach using equational abstraction

**Initial step.** We will start the refinement process with the abstraction that ignores the last two elements of a state. In order to take fully advantage of equational abstraction, we will express it using more than one sentence:

$\Delta$: $(x, y, m, n) = (x', y', m', n')$ `if` $x = x',\ y = y',\ x = 0,\ y = 0$
$(x, y, m, n) = (x', y', m', n')$ `if` $x = x',\ y = y',\ x + y = 1$
$(x, y, m, n) = (x', y', m', n')$ `if` $x = x',\ y = y',\ x = 1,\ y = 1$
$(x, y, m, n) = (x', y', m', n')$ `if` $x = x',\ y = y',\ x + y = 2,\ x : \texttt{nat02},\ y : \texttt{nat02}$
$(x, y, m, n) = (x', y', m', n')$ `if` $x = x',\ y = y',\ x = 2,\ y = 2$

The Kripke structure for this abstraction is depicted in Figure 5.3(a) (we have denoted the abstract states with the values of $x$ and $y$). We can easily produce a counterexample for $\phi$, let it be the path $(0, 0)$, $(1, 0)$, $(1, 1)$. Trying to unfold the counterexample in the concrete system, we get that it is spurious, that is, it has no correspondent in the concrete system. Then, as in [25], we search for an *abstract failure state* and we extract the set $D$ of *dead-end states* (states from which we have no transition to the abstract state that follows the failure state) and the set $B$ of *bad states* (states from which we have transitions to the abstract state that follows the failure state). Thus, we obtain $D = \{(0, 0, 16, 16)\}$ and $B = \{(0, 0, m, n) \mid m\%2 = 1\}$. To separate the set of dead-end states from the set of bad states, we refine using the atomic formula "$m\%2 = m'\%2$", that it is added only to the first sentence above because, this is the only sentence in $\Delta$ needed to prove that the states from $D \cup B$ are in the same abstraction state.

**First refinement step.** The Kripke structure for this abstraction is depicted in Figure 5.3(b). We have denoted states by couples whose elements are the values of $x$ and $y$, or by triples in which the first two elements are

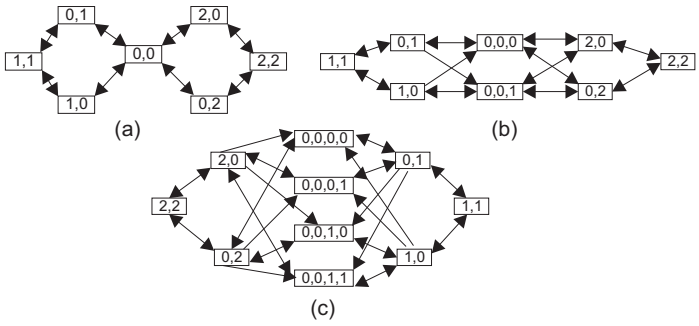


Figure 5.3: The abstraction for the initial, first and second step of refinement

the same and the last represents the value of $m\%2$. A counterexample for the required property is $(0,0,0)$, $(2,0)$, $(2,2)$. Trying to unfold this counterexample in the concrete system, we get that it is spurious, and obtain the set of dead-end states $D_1 = \{(0,0,16,16)\}$ and the set of bad states $B_1 = \{(0,0,m,n) \mid n\%2 = 1\}$. Consequently, we refine using the atomic formula "$n\%2 = n'\%2$" that it is added only to the first sentence of the abstraction because, this is the only sentence used to prove that the states from $D_1$ and $B_1$ are in the same abstraction state.

**Second refinement step.** The Kripke structure for this abstraction can be visualized in Figure 5.3(c). The abstract states are represented by couples and triples as before or by 4-tuples in which the first three elements are as in the case of triples and the last element is $n\%2$. Again, a counterexample can be found: $(0,0,0,0)$, $(0,1)$, $(1,1)$. Trying to unfold this counterexample in the concrete system, we get that it is spurious, and obtain the set of dead-end states $D_2 = \{(0,1,16,16)\}$ and the set of bad states $B_2 = \{(0,1,m,n) \mid m\%2 = 1\}$. Consequently, we refine using the atomic formula "$m\%2 = m'\%2$" that it is added, for the same reasons, only to the second sentence of the abstraction.

**Third refinement step.** The Kripke structure for this abstraction can be visualized in Figure 5.4(a) (the abstract states are denoted as before). Again, we can find a counterexample $(0,0,0,0)$, $(0,2)$, $(2,2)$ which by unfolding in the concrete system proves to be spurious. We obtain the set of dead-end states $D_3 = \{(0,2,16,16)\}$ and the set of bad states $B_3 = \{(0,2,m,n) \mid n\%2 = 1\}$.

Consequently, we refine using the atomic proposition "$n\%2 = n'\%2$" that it is added only to the fourth sentence of the abstraction.

**Final step.** The abstraction obtained after the third refinement step consists in the following set of sentences:

$(x,y,m,n) = (x',y',m',n')$ `if` $x = x', y = y', x = 0, y = 0, m\%2 = m'\%2,$

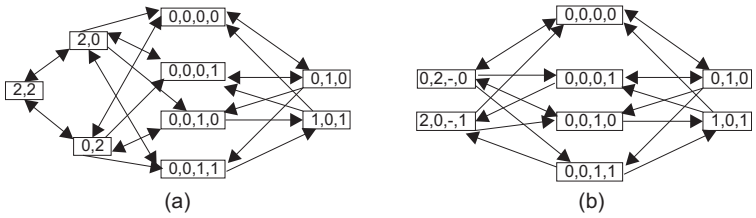


Figure 5.4: The abstraction for the third and fourth step of refinement

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad n\%2 = n'\%2 \\
(x,y,m,n) = (x',y',m',n') \texttt{ if } x = x',\ y = y',\ x+y = 1,\ m\%2 = m'\%2 \\
(x,y,m,n) = (x',y',m',n') \texttt{ if } x = x',\ y = y',\ x = 1,\ y = 1 \\
(x,y,m,n) = (x',y',m',n') \texttt{ if } x = x', y = y', x+y = 2, x : \texttt{nat02}, y : \texttt{nat02}, \\
\qquad\qquad\qquad\qquad n\%2 = n'\%2 \\
(x,y,m,n) = (x',y',m',n') \texttt{ if } x = x',\ y = y',\ x = 2,\ y = 2
\end{array}
$$

The Kripke structure for this abstraction can be visualized in Figure 5.4(b). Abstraction states are denoted as before, except for $(0, 2, -, 0)$ and $(2, 0, -, 1)$ where the first two elements represent the values of $x$ and $y$ and the fourth the value of $n\%2$ (the third component just say that $m\%2$ can have any value).

Now, we obtain that $\phi$ is true in the current abstraction and consequently, we obtain that $\phi$ is true in the initial model.

### The approach using predicate abstraction

**Initial step.** We will start the refinement process with the same abstraction that ignores the last two elements of a state. For this, we use the set of predicates $\mathcal{B} = \{P_x^1, P_x^2, P_y^1, P_y^2\}$, where $P_v^i$ is 1 in the states with $v = i$, for all $v \in \{x, y\}$ and $i \in \{1, 2\}$. The Kripke structure obtained is again, the one from Figure 5.3(a). We consider the same spurious counterexample: $(0, 0), (1, 0), (1, 1)$, for which we obtain the set of dead-end states $D = \{(0, 0, 16, 16)\}$ and the set of bad states $B = \{(0, 0, m, n) \mid m\%2 = 1\}$. In order to separate $D$ and $B$, we should refine by adding the predicate $Q((x, y, m, n)) = m\%2$.

**First refinement step.** The Kripke structure of the abstraction that we obtain after the initial step is the one from Figure 5.5(a). We have denoted abstract states by triples where the first two components are the values of $x$ and $y$, respectively and the third component is the truth value of $Q$.

A counterexample can be found: $(0, 0, 0), (0, 2, 0), (2, 2, 0)$ which by unfolding in the concrete system is proved to be spurious. We find the set of dead-end states $D_1 = \{(0, 2, 16, 16)\}$ and the set of bad states $B_1 =$

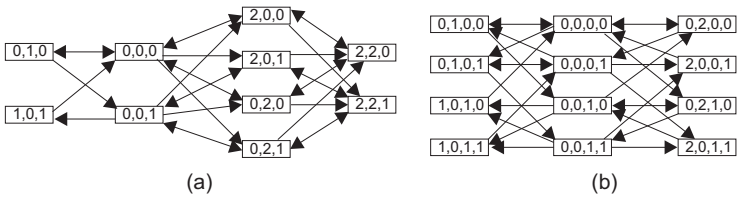


Figure 5.5: The abstraction for the first and second step of refinement

$\{(0, 2, m, n) \mid n\%2 = 1\}$. To separate them, we should add the predicate $S((x, y, m, n)) = n\%2$.

**Final refinement step** The Kripke structure for the current abstraction is the one from Figure 5.5(b). Now, the model checking procedure over the current abstraction returns 1 and consequently, $\phi$ is 1 in the concrete system.

### Comparison results

The first conclusion we draw when seeing the two proofs above is that CEGAR under equational abstraction obtained smaller models: the final model from CEGAR under predicate abstraction has 50% more states and 40% more transitions. The refinement using equational abstraction performed better because of the modularity of the system (the system consists in two modules that grant access to drink or food). When predicate abstraction discovers one predicate it splits all the abstraction states in two, although the predicate could have been relevant only for the abstraction states of a single module. For example, in the approach using equational abstraction, we didn't split the state $(0, 1, 0)$ in two abstraction states, as we did in the approach using predicate abstraction, depending on the value of $n$ because this value does not influence the transitions involving this state (similarly for the states $(1, 0, 1)$, $(0, 2, -, 0)$ and $(2, 0, -, 1)$).

Moreover, we can use the predicates that separate the set of dead-end states from the set of bad states, to derive the atomic formulas added in the case of equational abstraction. For example, from the predicate dicovered in the initial step $Q((x, y, m, n)) = m\%2$ we can derive the atomic formula $m\%2 = m'\%2$ discovered in the initial step of CEGAR under equational abstraction. This happened because the sentences of the abstraction have the form "$t = t'$ `if` $\mathcal{C}$", where $t$ and $t'$ are terms of kind `state`. Consequently, for CEGAR under equational abstraction we can use the already developed procedures that discover predicates for CEGAR under predicate abstraction. The single drawback of the proof using our refinement technique was that we had more refinement steps. However, this can not be proved to hold generally and, moreover, using other refinement procedures that eliminate

more counterexamples at the same time, we could improve this matter.

### 5.2.3 CEGAR under Equational Abstraction

We dedicate this section to the formalization of the CEGAR procedure exemplified in the previous section.

**Definition 5.2** Let $\mathcal{R} = (E, E_{AP}, \mathcal{E})$ be an $\Omega_D$ rewrite theory over a set of variables $X$. An abstraction of $\mathcal{R}$ is a set $\Delta$ of sentences over $\Omega_D$ and $X$.

The sentences from $\Delta$ have the role to induce a congruence on the equivalence classes of terms that represent states of the system. If $\mathcal{T}_{\Omega_D,E} = (T, \Sigma^T, \Pi_T)$ is the initial algebra in the class $Mod(E)$, then the set of states of the abstract system will be $T_{\texttt{state}}/\rho$, where $\rho$ is the congruence defined by:

$$[t]_{=_E} \ \rho \ [t']_{=_E} \text{ iff } E \cup \Delta \vdash t = t',$$

for any $t, t'$ terms of kind `state`.

Hence, to specify the abstract system induced by an abstraction as above, we just have to add the set of sentences $\Delta$ to $E$ and modify the sentences from $E_{AP}$ so the predicates in the abstract system under-approximate the ones in $\mathcal{R}$.

**Definition 5.3** Let $\mathcal{R}$ and $\Delta$ be as above. The *rewrite theory for the abstraction of* $\mathcal{R}$ *by* $\Delta$ is $\mathcal{R}_\Delta = (E \cup \Delta, E^\Delta_{AP}, \mathcal{E})$ such that for any $p \in \Sigma_L$:

- $E \cup \Delta \cup E^\Delta_{AP} \vdash t : s_p$ iff $(\forall t')(E \cup \Delta \vdash t = t' \Rightarrow E \cup E_{AP} \vdash t' : s_p)$.

The next result is straightforward and shows that abstractions of rewrite theories imply abstractions of corresponding Kripke structures.

**Theorem 5.2** Let $\mathcal{R}$, $\Delta$ and $\mathcal{R}_\Delta$ be as above. Also, consider the initial algebra $\mathcal{T}_{\Omega_D,E} = (T, \Sigma^T, \Pi_T)$ and the equivalence relation on $T_{\texttt{state}}$ given by:

$$[t]_{=_E} \ \rho \ [t']_{=_E} \text{ iff } E \cup \Delta \vdash t = t',$$

for any terms $t, t'$ of kind `state`. Then, $M(\mathcal{R}_\Delta)$ is the $(\alpha_R, \alpha_L)$-abstraction of $M(\mathcal{R})$ by $\rho$, where $\alpha_R(0) = \alpha_L(1) = \forall$ and $\alpha_R(1) = \alpha_L(0) = \exists^{\{0,1\}}$.

The theorem above also implies that the truth of $LTL_+$ formulas is preserved from the abstract system to the concrete one.

**Corollary 5.1** Let $\mathcal{R}$, $\Delta$ and $\mathcal{R}_\Delta$ be as above. Then, $\mathcal{I}_\mathcal{R}(\phi, [q]) = 1 \Rightarrow \mathcal{I}_\mathcal{R}(\phi, q) = 1$, for any $q$ state of $\mathcal{R}$.

The meaning of refinement in the context of equational abstraction can be formalized as follows.

**Definition 5.4** Let $\mathcal{R} = (E, E_{AP}, \mathcal{E})$ be a rewrite theory and $\Delta$, $\Delta'$ two abstractions of it. We say that $\Delta'$ refines $\Delta$ if

$$E \cup \Delta' \vdash t = t' \Rightarrow E \cup \Delta \vdash t = t',$$

for any terms $t, t'$ of kind `state`.

As we expected refinement implies that an abstract state of $M(\mathcal{R}_\Delta)$ may contain more abstract states of $M(\mathcal{R}'_\Delta)$. One of the advantages when using equational abstractions is that we have many alternatives in refining them: we can remove sentences, we can add atomic formulas to the conditions of the sentences or we can replace atomic formulas with stronger ones.

We present now the CEGAR procedure for equational abstractions of rewrite theories. The interesting part is the one that refines the abstraction, where instead of adding predicates, as in the case of CEGAR under predicate abstraction, we add atomic formulas to the conditions of the sentences that represent the current abstraction. We will remind CEGAR [25] and detail on the refinement part.

```
begin
  let R be a rewrite theory and q a state in M(R);
  let Δ be the initial abstraction;
  let φ be some property in LTL+ we want to prove;
  while true do
    x := modelcheck(R, Δ, φ, q);
    if x = 1 then
      return I_R(φ, q) = 1;
    else
      if isConcrete(counterexample(R, Δ, φ, q)) then
        return I_R(φ, q) = 0;
      else
        Δ := refine(Δ,counterexample(R, Δ, φ, q));
end
```

Above, `modelcheck`$(\mathcal{R}, \Delta, \phi, q)$ does model checking on $M(\mathcal{R}_\Delta)$ to verify $\phi$ in state $[q]$. If the output is 0, then `counterexample`$(\mathcal{R}, \Delta, \phi, q)$ returns a counterexample for $\phi$ in the abstraction. The function `isConcrete` checks if this counterexample is also a counterexample for $\phi$ in the concrete system. If not, the counterexample is called *spurious* and we have to refine the abstraction. This is done in the procedure `refine`, where we modify the set

of sentences $\Delta$ in order to remove from the abstraction the spurious counterexample. As we have already discussed, we adopt the approach from [25], and we search for the set $D$ of dead-end states and the set $B$ of bad states. Then, we discover atomic formulas that separate $B$ from $D$ and add them only to the conditions of the sentences from $\Delta$ that are necessary to prove that $B \cup D$ are in the same abstract state.

Now, we intend to give a formal idea about the fact that the procedure above constructs smaller models to be verified than CEGAR under predicate abstraction.

Predicate abstraction can be viewed as a particular case of equational abstraction. We suppose that the predicates used for abstraction are given by terms with possible values 0 and 1. A predicate abstraction induced by the set of predicates $\{p_1, \ldots, p_m\}$ can be expressed by an equational abstraction $\Delta_P = \{s_1, \ldots, s_n\}$ such that $s_i$ has the form

$$t_i = t_i' \text{ if } p_{i,1} = p_{i,1}', \ldots, p_{i,m} = p_{i,m}',$$

for any $1 \leq i \leq n$, where $t_i, t_i'$ are terms of kind `state`, $p_{i,j}$ is a version of the term representing $p_j$ that uses variables from $t_i$ and, for any term $v$, $v'$ is a term obtained from $v$ by replacing each variable $x$ with the primed version $x'$.

The refinement step in CEGAR under predicate abstraction will add a new predicate to the abstraction, which in the reformulation of the abstraction using membership sentences, means adding a new atomic formula $p_{i,m+1} = p_{i,m+1}'$ to each sentence $s_i$. The refinement step in the CEGAR under equational abstraction may add the same atomic formula $p_{i,m+1} = p_{i,m+1}'$ but, only to some of the sentences expressing the current abstraction. Therefore, if $\Delta_P'$ is the abstraction obtained in the first case and $\Delta_P''$ the abstraction obtained in the second case, we have that

$$E \cup \Delta_P'' \models t = t' \text{ implies } E \cup \Delta_P' \models t = t',$$

for any $t$, $t'$ terms of kind `state` ($E$ is the set of membership sentences that specify the data type for the set of states of the concrete system). The fact that in the abstract system obtained using $\Delta_P''$ we may have more equations $t = t'$ that hold, means that we may have fewer abstract states and the abstraction may be smaller.

# Index

# Bibliography


[1] P. E. Ammann and R. Sandhu. Extending the creation operation in the schematic protection model. In *Proceedings of the 6th Annual Computer Security Applications Conference*, pages 304–348, 1990.

[2] E. Astesiano, M. Broy, and G. Reggio. Algebraic specification of concurrent systems. In E. Astesiano, B. Krieg-Bruckner, and H.-J. Kreowski, editors, *IFIP WG 1.3 Book on Algebraic Foundations of System Specification.* Springer Verlag, 1999.

[3] T. Ball, B. Cook, S. Das, and S. K. Rajamani. Refining approximations in software predicate abstraction. In K. Jensen and A. Podelski, editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 2004.

[4] T. Ball, A. Podelski, and S. K. Rajamani. Boolean and cartesian abstraction for model checking C programs. In Margaria and Yi [83], pages 268–283.

[5] N. Belnap. A useful four-valued logic. In Donn and Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 30–56. Reidel, 1977.

[6] M. Bidoit and A. Boisseau. Algebraic abstractions. In M. Cerioli and G. Reggio, editors, *WADT*, volume 2267 of *Lecture Notes in Computer Science*, pages 21–47. Springer, 2001.

[7] L. Bolc and P. Borowik. *Many-valued Logics.* Springer-Verlag, 1992.

[8] A. Bouhoula, J. P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236(1-2):35–132, 2000.

[9] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In Halbwachs and Peled [61], pages 274–287.

[10] G. Bruns and P. Godefroid. Temporal logic query checking. In *LICS*, pages 409–417, 2001.

[11] G. Bruns and P. Godefroid. Model checking with multi-valued logics. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293. Springer, 2004.

[12] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[13] J. A. Brzozowski and C. J. H. Seger. A unified framework for race analysis of asynchronous networks. *Journal of the ACM*, 36(1):20–45, 1989.

[14] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: $10^{20}$ states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[15] D. R. Chase, M. N. Wegman, and F. K. Zadeck. Analysis of pointers and structures. In *PLDI*, pages 296–310, 1990.

[16] M. Chechik, B.Devereux, and S. M. Easterbrook. Implementing a multi-valued symbolic model checker. In Margaria and Yi [83], pages 404–419.

[17] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering Methodologies.*, 12(4):371–408, 2003.

[18] M. Chechik and W. Ding. Lightweight reasoning about program correctness. *Information Systems Frontiers*, 4(4):363–377, 2002.

[19] M. Chechik, A. Gurfinkel, and B. Devereux. $\chi$-check: A multi-valued model-checker. In E. Brinksma and K. G. Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 505–509. Springer, 2002.

[20] M. Chechik and W. MacCaull. CTL model-checking over logics with non-classical negations. In *ISMVL*, pages 293–. IEEE Computer Society, 2003.

[21] K. Cho and R. E. Bryant. Test pattern generation for sequential mos circuits by symbolic fault simulation. In *DAC*, pages 418–423, 1989.

[22] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In D. Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[23] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In Courcoubetis [31], pages 450–462.

[24] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the Futurebus+ cache coherence protocol. In D. Agnew, L. J. M. Claesen, and R. Camposano, editors, *CHDL*, volume A-32 of *IFIP Transactions*, pages 15–30. North-Holland, 1993.

[25] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[26] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[27] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[28] E. M. Clarke and R. P. Kurshan, editors. *Computer Aided Verification, 2nd International Workshop, CAV '90, New Brunswick, NJ, USA, June 18-21, 1990, Proceedings*, volume 531 of *Lecture Notes in Computer Science*. Springer, 1991.

[29] E. M. Clarke, D. E. Long, and K. L. McMillan. Compositional model checking. In *LICS*, pages 353–362. IEEE Computer Society, 1989.

[30] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.

[31] C. Courcoubetis, editor. *Computer Aided Verification, 5th International Conference, CAV '93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*. Springer, 1993.

[32] C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. In Clarke and Kurshan [28], pages 233–242.

[33] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977.

[34] D. Dams. *Abstract Interpretation and Partial Refinement for Model Checking*. PhD thesis, Technische Universitat Eindhoven, 1996.

[35] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transaction on Programming Languages and Systems*, 19(2):253–291, 1997.

[36] S. Das, D. L. Dill, and S. Park. Experience with predicate abstraction. In Halbwachs and Peled [61], pages 160–171.

[37] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[38] J. Dunn. A comparative study of various model-theoretic treatments of negation: a history of formal negation. In D. Gabbay and H. Wansing, editors, *What is Negation*. Kluwer Academic Publishers, 1999.

[39] S. M. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *ICSE*, pages 411–420. IEEE Computer Society, 2001.

[40] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. Springer-Verlag, 1985.

[41] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Module Specifications and Constraints*. Springer-Verlag, 1990.

[42] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. *Electronic Notes in Theoretical Computer Science*, 71, 2002.

[43] E. A. Emerson and A. P. Sistla. Symmetry and model checking. In Courcoubetis [31], pages 463–478.

[44] C. Enea. Unifying decidability results on protection systems using simulations. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust*, volume 3866 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2005.

[45] C. Enea and C. Dima. Abstractions of multi-agent systems. In H. D. Burkhard, G. Lindemann, R. Verbrugge, and L. Zsolt Varga, editors, *CEEMAS*, volume 4696 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2007.

[46] C. Enea and F. L. Tiplea. Abstractions of dynamic data types. *Acta Informatica*. submitted.

[47] C. Enea and F. L. Tiplea. Multi-valued abstractions. *Acta Informatica*. submitted.

[48] DeMillo et al., editor. *Foundations of Secure Computation*. Academic Press, 1978.

[49] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[50] J. C. Fernandez, C. Jard, T. Jéron, and C. Viho. Using on-the-fly verification techniques for the generation of test suites. In R. Alur and T. A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 1996.

[51] M. Fitting. Many-valued modal logics I. *Fundamenta Informaticae*, 15(3-4):235–254, 1991.

[52] M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, 17(1-2):55–73, 1992.

[53] B. R. Gaines. Logical foundations for database systems. *International Journal of Man-Machine Studies*, 11(4):481–500, 1979.

[54] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol, Specification, Testing and Verification*, pages 3–18, 1995.

[55] M. Ginsberg. Multivalued logics. a uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[56] P. Godefroid and D. Pirottin. Refining dependencies improves partial-order verification methods (extended abstract). In Courcoubetis [31], pages 438–449.

[57] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[58] S. Graf and B. Steffen. Compositional minimization of finite state systems. In Clarke and Kurshan [28], pages 186–196.

[59] O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.

[60] A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In R. M. Amadio and D. Lugiez, editors, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2003.

[61] N. Halbwachs and D. Peled, editors. *Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Proceedings*, volume 1633 of *Lecture Notes in Computer Science*. Springer, 1999.

[62] M. A. Harrison and W. L. Ruzzo. Monotonic protection systems. In et al. [48].

[63] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of ACM*, 19(8):461–471, 1976.

[64] J. P. Hayes. Pseudo-boolean logic circuits. *IEEE Transactions on Computers*, 35(7):602–612, 1986.

[65] S. Hazelhurst. *Compositional Model Checking of Partially Ordered State Spaces*. PhD thesis, University of British Columbia, 1996.

[66] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *POPL*, pages 58–70, 2002.

[67] S. Horwitz, P. Pfeiffer, and T. W. Reps. Dependence analysis for pointer variables. In *PLDI*, pages 28–40, 1989.

[68] C. N. Ip and D. L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1/2):41–75, 1996.

[69] N. D. Jones and S. Muchnick. Flow analysis and optimization of LISP-like structures. In S. Muchnick and N. D. Jones, editors, *Program Flow Analysis: Theory and Applications*, pages 102–131. Prentice-Hall, 1981.

[70] N. D. Jones and S. S. Muchnick. A flexible approach to interprocedural data flow analysis and programs with recursive data structures. In *POPL*, pages 66–74, 1982.

[71] S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, New York, 1952.

[72] B. Konikowska and W. Penczek. Reducing model checking from multi-valued CTL* to CTL*. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR2002)*, volume LNCS 2421, pages 226–239, 2002.

[73] B. Konikowska and W. Penczek. On designated values in multi-valued CTL* model checking. *Fundamenta Informaticae*, 60(1-4):211–224, 2004.

[74] B. Konikowska and W. Penczek. Model checking for multivalued logic of knowledge and time. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *AAMAS*, pages 169–176. ACM, 2006.

[75] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[76] R. P. Kurshan. *Computer-Aided Verification of Coordinating Processes.* Princeton University Press, 1994.

[77] J. R. Larus and P. N. Hilfinger. Detecting conflicts between structure accesses. In *PLDI*, pages 21–34, 1988.

[78] R. J. Lipton and L. Snyder. A linear time algorithm for deciding subject security. *Journal of the ACM*, 24(3):455–464, 1977.

[79] R. J. Lipton and L. Snyder. On synchronization and security. In et al. [48].

[80] A. Loginov, T. Reps, and M. Sagiv. Abstraction refinement via inductive learning. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), LNCS 3576*, pages 519–533, Edinburgh, Scotland, UK, 2005.

[81] Z. Lotfi. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[82] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems. Specification.* Springer-Verlag, 1992.

[83] T. Margaria and W. Yi, editors. *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and*

*Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2031 of *Lecture Notes in Computer Science*. Springer, 2001.

[84] N. Marti-Oliet, J. Meseguer, and M. Palomino. Algebraic simulations. 2005. manuscript: http://maude.cs.uiuc.edu/papers/.

[85] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic, 1993.

[86] K. L. McMillan. Verification of infinite state systems by compositional model checking. In L. Pierre and T. Kropf, editors, *CHARME*, volume 1703 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 1999.

[87] K. L. McMillan and N. Amla. Automatic abstraction without counterexamples. In *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03), LNCS 2619*, pages 2–17, Warsaw, Poland, 2003.

[88] K. L. McMillan and J. Schwalbe. Formal verification of the Gigamax cache consistency protocol. In N. Suzuki, editor, *Shared Memory Multiprocessing.* The MIT Press, 1992.

[89] K. Meinke and J. V. Tucker. Universal algebra. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science vol. 1*, pages 189–411. Oxford University Press, 1993.

[90] J. Meseguer. Conditioned rewriting logic as a united model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[91] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proceedings of WADT 97*, volume LNCS 1376, pages 18–61, 1998.

[92] J. Meseguer, M. Palomino, and N. Martí-Oliet. Equational abstractions. In *Proceedings of CADE 03*, volume LNCS 1376, pages 2–16, 2003.

[93] R. S. Michalski. Variable-valued logic and its applications to pattern recognition and machine learning. *In D. C. Rine, editor, Computer Science and Multiple-Valued Logic: Theory and Applications*, page 506534, 1977.

[94] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489, 1971.

[95] J. Mitchell. *Foundations of Programming Languages*. The MIT Press, 1996.

[96] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.

[97] K. R. O'Neill and J. Y. Halpern. Secrecy in multiagent systems. *CoRR*, cs.CR/0307057, 2003.

[98] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[99] D. Peled. Combining partial order reductions with on-the-fly model-checking. In D. L. Dill, editor, *CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 1994.

[100] J. Plevyak, A. A. Chien, and V. Karamcheti. Analysis of dynamic structures for efficient parallel execution. In U. Banerjee, D. Gelernter, A. Nicolau, and D. A. Padua, editors, *LCPC*, volume 768 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1993.

[101] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglini and U. Montanari, editors, *Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982.

[102] H. Rasiowa. *An Algebraic Approach to Non-Classical Logics. Studies in Logic and the Foundations of Mathematics.* Amsterdam:North Holland, 1978.

[103] S. Sagiv, T. W. Reps, and R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. *ACM Transactions on Programming Languages and Systems*, 20(1):1–50, 1998.

[104] S. Sagiv, T. W. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems*, 24(3):217–298, 2002.

[105] H. Saidi and N. Shankar. Abstract and model check while you prove. In *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, pages 443–454, Trento, Italy, 1999.

[106] R. Sandhu. The typed access matrix model. In *Proceedings of the 13th IEEE Symposium on Research in Security and Privacy*, pages 122–136, 1992.

[107] R. S. Sandhu. The schematic protection model: its definition and analysis for acyclic attenuating schemes. *Journal of the ACM*, 35(2):404–432, 1988.

[108] T. Sasao and J. T. Butler. A method to represent multiple-output switching functions by using multi-valued decision diagrams. In *Proceedings of the IEEE International Symposium on Multiple-Valued Logic*, pages 248–254, 1996.

[109] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton. Algorithms for discrete function manipulation. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 92–95, 1990.

[110] J. Stransky. A lattice for abstract interpretation of dynamic LISP-like structures. *Information and Computation*, 101(1):70–102, 1992.

[111] F. L. Tiplea and C. Enea. Abstractions of data types. *Acta Informatica*, 42(8-9):639–671, 2006.

[112] A. Valmari. A stubborn attack on state explosion. In Clarke and Kurshan [28], pages 156–165.

[113] W. Visser, S. Park, and J. Penix. Using predicate abstraction to reduce object-oriented programs for model checking. In M. P. E. Heimdahl, editor, *FMSP*, pages 3–182. ACM, 2000.

[114] E. Y. B. Wang. *Analysis of Recursive Types in an Imperative Language*. PhD thesis, University of California, Berkeley, 1994.