Reasoning about Composition using Property Transformers and their Conjugates*

Michel Charpentier[1] and K. Mani Chandy[2]

[1] University of New Hampshire
Computer Science Department
email: charpov@cs.unh.edu

[2] California Institute of Technology
Computer Science Department
e-mail: mani@cs.caltech.edu

Abstract. Compositional design is concerned with both constructing systems by composing components and with deconstructing systems into proposed sets of components. In bottom-up design, engineers prove system properties given properties of components and a compositional structure. In top-down design, they propose properties of components and a compositional structure given system properties. In this paper we show how the theory of predicate transformers, which has been used so successfully in sequential programming, can be applied to compositional design of systems. The rules of composition we study are more general than the rules employed in sequential programming, and the systems we study are not limited to programs. We exploit theorems about weakest and strongest solutions to equations to obtain a collection of useful predicate transformers, and then we exploit the theory of conjugate transformers to obtain more useful transformers. We show how these transformers are useful for both bottom-up and top-down design.

1 Motivation

1.1 Composition and Compositional Properties

Composition is the most fundamental operation in design. Designers of space vehicles, buildings and programs have common concerns: How to compose systems from components and how to partition systems into components. Compositional design offers the hope of managing complexity by avoiding unnecessary detail: systems designers prove properties of systems given properties, but not detailed implementations, of components.

We introduce an informal concept of *compositional properties* to motivate our exploration, and define terms precisely later. Compositional properties are those classes of properties that allow designers to deduce system properties from

* *IFIP International Conference on Theoretical Computer Science (TCS'2000)*, August 2000, LNCS 1872, pages 580–595. © Springer-Verlag.

component properties using simple rules. For example, mass is a compositional property because the mass of a system can be deduced in a simple way from the masses of components: the system mass is the sum of component masses. By contrast, temperature does not appear to be a compositional property because the temperature of a system depends in very complex ways on the shapes, masses, insulation properties, power consumption, locations of the components, etc.

Designers have to compute properties of composed systems given properties of components, whether the properties are compositional or not. The challenge is to develop theories that help designers prove system properties they need from component properties.

In this paper, we restrict ourselves to properties that are predicates on systems. We explore properties that are compositional in the following sense: we can prove that a property holds for a system given that the property holds for some or all of its components. In later papers we propose to explore other kinds of compositional properties.

In our paper, systems are abstract entities. They are not necessarily programs and they may not have "states" or "computations." We consider composition operators that have certain algebraic properties, such as associativity, and we explore theorems that are derived solely from these properties. Our goal is to explore composition in the abstract as opposed to studying how composition is used in constructing specific kinds of systems.

The simplest rules are those that establish that a property X holds for a system given that (i) property X holds for at least one component, or (ii) property X holds for all components. Therefore, in this paper, we restrict attention to two kinds of compositional properties: *existential properties* and *universal properties.* A property is an existential property exactly when, for all systems, a system has the property if there exists a component of the system that has the property. A property is a universal property exactly when, for all systems, a system has the property if all components of the system have the property.

1.2 An Introduction to Property Transformers for Composition

We motivate our exploration of predicate transformers by a few examples, and then develop the theory.

Questions about the Weakest Existential Transformer. Consider the following specification S for a component F: All systems that have F as a component must have a property X.

We postulate that any system is a component of itself. Since F is a component of F it follows that F itself must have property X. If X is an existential property, then from the definition of existential properties it follows that X holds for all systems that contain F as a component. Therefore, if X is an existential property, the given specification S is equivalent to the simpler specification: X holds in F. What if property X is not existential?

Suppose we can define a predicate transformer $\mathbb{WE}$ where $\mathbb{WE}.X$ is an existential property stronger than X. If we can demonstrate that component F

has existential property $\mathbb{WE}.X$, then any system that includes component F also has property $\mathbb{WE}.X$, and therefore also enjoys the weaker property X. Let S' be the following specification of F: $\mathbb{WE}.X$ holds in F. From the above argument, it follows that specification S' is stronger than specification S.

Examples of the kinds of questions that we wish to explore are the following. For a given property X, is there a *weakest* existential property at least as strong as X? And, if this weakest property exists and we define $\mathbb{WE}.X$ to be this property, then are specifications S' and S equivalent?

Questions about the Strongest Existential Transformer. Next, consider a dual set of questions. Given that system F has property X, what properties can we deduce about all systems that have F as a component? If X is existential, then all systems that have F as a component also have property X. But, what if X is not existential?

Suppose we can define a predicate transformer $\mathbb{SE}$ where $\mathbb{SE}.X$ is an existential property weaker than X. If F has property X then it also has the weaker property $\mathbb{SE}.X$ and, since $\mathbb{SE}.X$ is existential, all systems that contain F as a component also satisfy $\mathbb{SE}.X$. The obvious question to explore is: Can we define $\mathbb{SE}.X$ to be the *strongest* existential property weaker than X? And if we can, is $\mathbb{SE}.X$ the strongest property that holds for all systems that contain F as a component?

Questions about the Conjugate Weakest Existential Transformer. Now, consider an engineer designing a system top down. The designer is given the specification that the system must have property X. The designer asks the question: Can I restrict myself to considering only those components that have a property Y? In other words, can we prove that any system that contains a component that does *not* have property Y also does *not* have property X? We will show that the conjugate of the weakest existential transformer is helpful in answering this question.

Questions about Universal Transformers. We also consider the analogous case for universal properties. We explore the following question: Is there a weakest property Y such that, if all components of any system have property Y, then the system has property X? If X is a universal property and all components have property X then the system itself has property X. So, since Y must be at least as strong as X, Y is the same as X. What if X is not universal?

We can introduce a predicate transformer $\mathbb{WU}$ with the requirement that $\mathbb{WU}.X$ is universal and stronger than X. If we can then prove that all components of a system have property $\mathbb{WU}.X$ then we can conclude that the system enjoys this property and hence also enjoys the weaker property X.

Can we require that $\mathbb{WU}.X$ be the *weakest* universal property stronger than X? We can show that we cannot do so because there does not exist, in general, a weakest universal property stronger than X. What are good ways of defining $\mathbb{WU}$, then? We do not have answers to this question.

In the main body of this paper, we explore similar questions about strongest universal transformers and their conjugates.

2 Terminology and Notations

2.1 Composition

We postulate a composition operator denoted by $\circ$ and we restrict ourselves to systems built from a finite number of applications of that operator. We assume that $\circ$ is associative. We do not assume other properties such as symmetry or idempotency. We do not interpret systems, and we do not consider how systems are constructed by composing elemental or atomic systems. All that is relevant in this paper is that systems can be composed to obtain systems.

We postulate the existence of a binary relation $\surd$ and we only consider systems $F{\circ}G$ for those components F and G for which $F\surd G$ holds. We assume that $F\surd G$ denotes that F can be composed with G (in that order).

We assume that if the system $F{\circ}G{\circ}H$ can be constructed, then it can be constructed by first composing F with G and then composing the resulting system with H, or by first composing G with H and then composing the resulting system with F on the left. Specifically, we assume the following property of $\surd$, for any F, G and H:

$$F\surd G \wedge (F{\circ}G)\surd H \;\equiv\; G\surd H \wedge F\surd(G{\circ}H) \;.$$

We assume the existence of a UNIT component that satisfies the following axiom for all systems F:

$$\mathit{UNIT}\surd F \;\wedge\; F\surd \mathit{UNIT} \;\wedge\; (\mathit{UNIT}{\circ}F = F{\circ}\mathit{UNIT} = F) \;. \tag{1}$$

For some theorems, we need the additional axiom that the unit system cannot have non-unit components:

$$(F{\circ}G = \mathit{UNIT}) \;\equiv\; (F = \mathit{UNIT}) \wedge (G = \mathit{UNIT}) \;. \tag{2}$$

Most results presented in this paper do not require this additional axiom. However, some results do. These results are marked with a $^{\circledast}$ sign.

2.2 Membership Relation

We introduce a specific notation to denote that a system F is part of a system G:

$$F \lhd G \;\triangleq\; \langle \exists H, K : H\surd F \wedge H{\circ}F\surd K : G = H{\circ}F{\circ}K \rangle \;.$$

Note that, because of the axiom (1) on the UNIT element, $\lhd$ is a reflexive operator and $\mathit{UNIT} \lhd F$ is true for any F.

2.3 Properties and Specifications

Properties are point-wise predicates on systems. We treat a property as boolean-valued function with a single argument of type system. We use dot notation to denote function application, as in $f.x$ denotes the application of function f to x.

Therefore, for any property X and any system F, the notation $X.F$ denotes the boolean: system F has property X. Following [13], we use square brackets to denote that a predicate is "everywhere true". For a property X, $[X]$ is the boolean: property X holds in all systems.

We introduce two properties that are specific to the *UNIT* component, $UNIT_=$ ("to be the *UNIT*") and its negation $UNIT_{\neq}$ ("to be different from the *UNIT*"):

$$UNIT_= \ . \ F \ \triangleq \ (F = UNIT) \qquad \text{and} \qquad UNIT_{\neq} \ . \ F \ \triangleq \ (F \neq UNIT) \ .$$

2.4 Bags of Colored Balls

In order to illustrate the ideas presented in this paper, we use a simple model of components. In this model, systems are bags of colored balls. Composition corresponds to bag-union and the *UNIT* element is the empty bag. Note that this bag model satisfies axiom (2). Therefore, properties marked with $^{\circledast}$ can be used when reasoning on this model.

Bags can always be composed (i.e., the relation $\surd$ is always *true*) and composition is symmetric (Abelian monoid), but we do not rely on these additional properties of the model for they may not be true of more interesting models. For instance, sequential composition of programs is not symmetric and parallel composition of processes may not always be possible (for example, if a process references a local variable from another process).

3 Existential and Universal Properties

3.1 Existential Properties

A property X is *existential* (denoted by the boolean $exist.X$) if and only if X holds in all systems that contain a component that satisfies X:

$$exist.X \ \triangleq \ \langle \forall F, G : F \surd G : X.F \vee X.G \Rightarrow X \ . \ F \circ G \rangle \ .$$

The following results can be proved about existential properties:

$$\begin{gathered} exist.X \wedge exist.Y \ \Rightarrow \ exist.(X \wedge Y), \\ exist.X \wedge exist.Y \ \Rightarrow \ exist.(X \vee Y), \\ exist.X \Rightarrow (X \ . \ UNIT \equiv [X]), \\ exist.UNIT_{\neq} \ ^{\circledast} \ . \end{gathered}$$

We define a function *guarantees*, from pairs of properties to properties. The property X *guarantees* Y holds for a system F if and only if, for all systems G that contain F as a component, if X holds for G then Y also holds for G:

$$X \ guarantees \ Y \ . \ F \ \triangleq \ \langle \forall G : F \lhd G : X.G \Rightarrow Y.G \rangle \ .$$

Proposition 1 *For all properties X and Y,*

$$exist.(X \ guarantees \ Y) \ .$$

Thus, *guarantees* provides us with a systematic way of building existential properties from other kinds of properties. The existential properties described with *guarantees* have been used successfully in compositional specifications and proofs of distributed systems [7].

3.2 Universal Properties

A property X is *universal* (denoted by the boolean $univ.X$) if and only if X is true in all systems built from components all of which satisfy X:

$$univ.X \ \triangleq \ \langle \forall F, G : F \surd G : X.F \wedge X.G \Rightarrow X \ . \ F \circ G \rangle \ .$$

From the definitions of *exist* and *univ*, any existential property is also universal:

$$exist.X \Rightarrow univ.X \ .$$

Moreover, the following properties can be proved:

$$\begin{aligned} univ.X \wedge univ.Y \ &\Rightarrow \ univ.(X \wedge Y), \\ univ.X \wedge exist.Y \ &\Rightarrow \ univ.(X \vee Y), \\ univ.UNIT_{=} \ . \end{aligned}$$

3.3 All-components and Some-component Properties

We define two additional forms of composition as the duals of existential and universal composition. A property is *all-components* if and only if its negation is existential:

$$all\text{-}c.X \ \triangleq \ exist.(\neg X) \ .$$

Unfolding the definition of *exist*, we find that a property is an all-components property if and only if, when it holds for any system, it holds for all components of that system:

$$all\text{-}c.X \ \equiv \ \langle \forall F, G : F \surd G : X \ . \ F \circ G \Rightarrow X.F \wedge F.G \rangle \ .$$

In the same way, we define *some-component* properties as the duals of universal properties:

$$some\text{-}c.X \ \triangleq \ univ.(\neg X) \ .$$

A property is a some-component property if and only if, when it holds for any system, it holds for at least one component of that system:

$$some\text{-}c.X \ \equiv \ \langle \forall F, G : F \surd G : X \ . \ F \circ G \Rightarrow X.F \vee F.G \rangle \ .$$

Existential and universal properties are used in bottom-up design. Their dual are used in top-down design.

- **Bottom-up.** Given components that have existential or universal properties, designers can deduce properties of systems composed from these components using very simple rules.
- **Top-down.** Given that a system should satisfy an all-components property, designers know that they must design all components to also satisfy that property. Likewise, given that a system should have a some-component property, designers know that they must design at least one component to have that property.

Of course, systems may be specified in terms of properties that are not existential, universal, all-components or some-component. However, as shown in the next sections, any property can be related in a systematic way to some properties that have one of these characteristics.

3.4 Examples of Properties

In the bags of colored balls model, the following properties are examples of existential, universal, all-components and some-component properties:

$$\begin{aligned}
&\mathit{exist} \;.\; \text{(at least one blue ball in the bag)} && (3)\\
&\mathit{exist} \;.\; \text{(at least two balls of different colors in the bag)} &&\\
&\mathit{univ} \;.\; \text{(no ball in the bag is blue)} && (4)\\
&\mathit{univ} \;.\; \text{(all the balls in the bag are red, or at least two are blue)} &&\\
&\mathit{all\text{-}c} \;.\; \text{(all the balls in the bag are blue, or all the balls are red)} &&\\
&\mathit{all\text{-}c} \;.\; \text{(there are at most three balls in the bag)} &&\\
&\mathit{some\text{-}c} \;.\; \text{(the bag contains more blues balls than red balls)} && (5)\\
&\mathit{some\text{-}c} \;.\; \text{(exactly one ball in the bag is red)} &&
\end{aligned}$$

Note that (3) is also some-component, (4) is also all-components and (5) is also universal. All other example properties only have the stated characteristic (besides the fact that any existential property is universal and any all-components property is some-component).

4 Property Transformers for Composition

4.1 Extreme Solutions of Equations in Predicates

As claimed in sect. 3, conjunctions of universal properties are universal and conjunctions and disjunctions of existential properties are existential[1]. An equation in predicates has a weakest solution if and only if the disjunction of all solutions is itself a solution, and in this case the weakest solution is that disjunction. Likewise, an equation in predicates has a strongest solution if and only if the conjunction of all solutions is itself a solution, and in this case the strongest solution is that conjunction [13].

[1] In sect. 3, junctivity is stated finitely for convenience, but actually holds for an infinite number of predicates.

4.2 The Property Transformer WE

We consider the following equation in Z, parametrized by predicate X:

$$Z : [Z \Rightarrow X] \wedge exist.Z \ . \tag{6}$$

A property Z is solution of (6) if and only if it is existential and stronger than X. Since disjunctions of existential properties are existential, equation (6) has a weakest solution, and we denote that weakest solution by $\mathrm{WE}.X$:

$$\mathrm{WE}.X \ \triangleq \ \langle \exists Z : [Z \Rightarrow X] \wedge exist.Z : Z \rangle \ .$$

For any property X, $\mathrm{WE}.X$ is the weakest existential property stronger than X.

Suppose we wish to design a system F so that any system that has F as a component enjoys property X. What properties must system F have? The next theorem tells us that the necessary and sufficient specification of such a component F is that it satisfies $\mathrm{WE}.X$.

Proposition 2 $\mathrm{WE}.X$ *is the weakest property of a component that ensures that any system that contains that component will enjoy property* X*:*

$$\langle \forall F, G : F \lhd G : Y.F \Rightarrow X.G \rangle \ \equiv \ [Y \Rightarrow \mathrm{WE}.X] \ .$$

This is a consequence of the following proposition that states that, for a component, to bring the property X to any system containing the component, or to satisfy $\mathrm{WE}.X$ are equivalent.

Proposition 3 $$\mathrm{WE}.X \ . \ F \ \equiv \ \langle \forall G : F \lhd G : X.G \rangle \ .$$

Proposition 3 is proved in [8]. The following elementary properties hold as well:

$$[\mathrm{WE}.X \Rightarrow X],$$
$$exist.X \ \equiv \ [\mathrm{WE}.X \equiv X],$$
$$[\mathrm{WE}.(X \wedge Y) \ \equiv \ \mathrm{WE}.X \wedge \mathrm{WE}.Y], \tag{7}$$
$$[X \Rightarrow Y] \Rightarrow [\mathrm{WE}.X \Rightarrow \mathrm{WE}.Y], \tag{8}$$
$$[\mathrm{WE}.X] \equiv [X] \ .$$

Property (7) states that the transformer WE is conjunctive. It can be shown that WE is *not* disjunctive. Property (8) expresses that WE is monotonic.

The property transformer WE can be used in two ways. Firstly, it provides us with an abstract way of expressing that a component, all by itself, ensures a system property X. Moreover, proposition 3 can be used to derive proof rules for properties of the form $\mathrm{WE}.X$ for a specific formalism. For instance, proof rules for UNITY logic are used in the correctness proof of a distributed system in [7]. Secondly, properties such as (7) and (8) can be used to reason about existential properties in general, without the inconvenience of a quantification over components. For instance, using proposition 3, it is easy to see that

$$[X \ guarantees \ Y \ \equiv \ \mathrm{WE}.(X \Rightarrow Y)] \ .$$

Then, it is easier to reason on WE to deduce properties of *guarantees* (such as existentiality or transitivity).

4.3 The Property Transformer $\mathbb{SE}$

The property transformer $\mathbb{WE}$ was defined using the fact that disjunctions of existential properties are existential. Because conjunctions of existential properties also are existential, the equation $Z : [X \Rightarrow Z] \wedge \mathit{exist}.Z$ has a strongest solution $\mathbb{SE}.X$:

$$\mathbb{SE}.X \triangleq \langle \forall Z : [X \Rightarrow Z] \wedge \mathit{exist}.Z : Z \rangle \ .$$

For any property X, $\mathbb{SE}.X$ is the strongest existential property weaker than X. Among interesting properties concerning $\mathbb{SE}$, we can prove:

$$\begin{gathered}
[X \Rightarrow \mathbb{SE}.X], \\
\mathit{exist}.X \ \equiv \ [\mathbb{SE}.X \equiv X], \\
[\mathbb{SE}.(X \vee Y) \ \equiv \ \mathbb{SE}.X \vee \mathbb{SE}.Y], \\
[X \Rightarrow Y] \Rightarrow [\mathbb{SE}.X \Rightarrow \mathbb{SE}.Y], \\
[\mathbb{SE}.X] \equiv (X \ . \ \mathit{UNIT})^{\circledast} \ .
\end{gathered} \tag{9}$$

Property (9) has an interesting intuitive explanation. $\mathbb{SE}.X$ holds for all systems that have at least one component that satisfies X. Since all systems have UNIT as a component, for all properties X that hold for UNIT, all systems have property $\mathbb{SE}.X$. So, as expected, designers do not get useful information from knowing that UNIT is a component of their systems.

Proposition 4 $\mathbb{SE}.X$ *is the strongest property that can be deduced of any system built from components, one of which, at least, satisfies* X*:*

$$\langle \forall F, G : F \lhd G : X.F \Rightarrow Y.G \rangle \ \equiv \ [\mathbb{SE}.X \Rightarrow Y] \ .$$

4.4 The Property Transformer $\mathbb{SU}$

From sect. 3, we know that conjunctions of universal properties are universal. However, disjunctions of universal properties are not always universal. Therefore, we cannot define a transformer $\mathbb{WU}$ in the same way as we defined $\mathbb{WE}$. Actually, it can be shown [8] that some properties do not have a weakest universal property stronger than them. Nevertheless, because conjunctions of universal properties are universal, the equation $Z : [X \Rightarrow Z] \wedge \mathit{univ}.Z$ has a strongest solution $\mathbb{SU}.X$:

$$\mathbb{SU}.X \triangleq \langle \forall Z : [X \Rightarrow Z] \wedge \mathit{univ}.Z : Z \rangle \ .$$

For any property X, $\mathbb{SU}.X$ is the strongest universal property weaker than X. Because $\mathbb{SE}.X$ is universal (since it is existential) and weaker than X, we can deduce:

$$[\mathbb{SU}.X \Rightarrow \mathbb{SE}.X] \ .$$

Among interesting properties concerning $\mathbb{SU}$, we can prove:

$$
\begin{aligned}
&[X \Rightarrow \mathbb{SU}.X], \\
univ.X \;\equiv\; &[\mathbb{SU}.X \equiv X], \\
[X \Rightarrow Y] \Rightarrow &[\mathbb{SU}.X \Rightarrow \mathbb{SU}.Y], \\
[\mathbb{SU}.X] \Rightarrow &(X \ . \ UNIT)^{\circledast} \ . \qquad (10)
\end{aligned}
$$

Note that property (10) is only an implication because $\mathbb{SU}.X$ is, in general, strictly stronger than $\mathbb{SE}.X$. Also, $\mathbb{SU}$ is monotonic but neither conjunctive, nor disjunctive.

If all components of a system satisfy X, they also satisfy $\mathbb{SU}.X$ and, because $\mathbb{SU}.X$ is universal, the whole system satisfies $\mathbb{SU}.X$. Moreover, $\mathbb{SU}.X$ is the strongest property that can be deduced this way.

Proposition 5 $\mathbb{SU}.X$ *is the strongest property that can be deduced of any system built from components that all satisfy* X*:*

$$
\begin{aligned}
&\langle \forall F, G, H, \cdots : F\surd G \wedge F{\circ}G\surd H \wedge F{\circ}G{\circ}H\surd \cdots : \\
&\qquad X.F \wedge X.G \wedge X.H \wedge \cdots \Rightarrow Y \ . \ F{\circ}G{\circ}H{\circ}\cdots\rangle \\
&\qquad\qquad \equiv [\mathbb{SU}.X \Rightarrow Y] \ .
\end{aligned}
$$

4.5 Example

We consider the following question: What must be proved on a bag of balls to ensure that any system that contains that bag, if it contains at least one red ball, contains at least two balls of different colors? Formally, the specification of the component is that any system that contains that component satisfies:

$$\text{(at least 1 red ball)} \Rightarrow \text{(at least 2 colors)} \ .$$

Then, from proposition 2, we know that the specification for the component is:

$$\mathbb{WE}.(\text{(at least 1 red ball)} \Rightarrow \text{(at least 2 colors)}) \ . \qquad (11)$$

In this section, we show how an explicit formulation of property (11) can be calculated. The calculation relies on the following proposition, proved in [8]:

Proposition 6 $\qquad [X \equiv UNIT_{\neq}] \vee [\mathbb{WE}.(UNIT_{=} \vee X) \equiv \mathbb{WE}.X]^{\circledast}$.

Then, we can calculate an equivalent formulation of (11):

$$
\begin{aligned}
&\mathbb{WE}.(\text{(at least 1 red ball)} \Rightarrow \text{(at least 2 colors)}) \\
= \;&\{\text{Predicate calculus}\} \\
&\mathbb{WE}.(UNIT_{=} \vee \text{(at least 1 non red ball)}) \\
= \;&\left\{\begin{array}{l}\text{Assume there exist red balls, hence} \\ \neg[\text{(at least 1 non red ball)} \equiv UNIT_{\neq}]\text{, apply prop. 6}\end{array}\right\}
\end{aligned}
$$

$$\begin{aligned} & \mathbb{WE}.(\text{at least 1 non red ball}) \\ = & \ \{\textit{exist}.(\text{at least 1 non red ball})\} \\ & (\text{at least 1 non red ball}) \ . \end{aligned}$$

Therefore, we deduce that the necessary and sufficient specification of the component is that it should contain at least one non-red ball. In other words, to ensure that any system that uses F as a component will have at least two balls of different colors provided it has at least one red ball, it is sufficient and necessary that F contains at least one non-red ball. This is consistent with our intuition. However, instead of guessing the desired property of F and then prove that it is both necessary and sufficient, we have *calculated* the property. This provides us with both the property and the proof at the same time, and avoids dealing explicitly with the universal quantification over components.

5 Conjugates of Property Transformers

Any predicate transformers has a unique *conjugate*. Duality allows us to easily deduce properties of a predicate transformer from properties of its conjugate. In this section, we focus on the conjugates of $\mathbb{WE}$, $\mathbb{SE}$ and $\mathbb{SU}$.

5.1 Conjugate of a Predicate Transformer

Let f be a predicate transformer. Its conjugate, denoted by f^* is defined by [13]:

$$f^*.X \ \triangleq\ \neg f.(\neg X) \ .$$

Duality provides us with a way of deducing properties of f^* from properties of f, and vice-versa. For instance, f is monotonic iff f^* is monotonic. In the same way, f is conjunctive (resp. disjunctive) iff f^* is disjunctive (resp. conjunctive).

5.2 The Property Transformer $\mathbb{WE}^*$

We define $\mathbb{WE}^*$ as the conjugate of the property transformer $\mathbb{WE}$:

$$\mathbb{WE}^*.X \ \triangleq\ \neg\mathbb{WE}.(\neg X) \ .$$

Because the dual of existential properties are all-components properties, we can easily deduce that $\mathbb{WE}^*.X$ is the strongest all-components property weaker then property X. Moreover, applying the duality principle to proposition 2, we can deduce the following proposition.

Proposition 7 *$\mathbb{WE}^*.X$ is the strongest property that can be deduced of all components of any system that satisfies X:*

$$\langle \forall F, G : F \lhd G : X.G \Rightarrow Y.F \rangle \ \equiv\ [\mathbb{WE}^*.X \Rightarrow Y] \ .$$

Duality can also be applied to the basic properties of $\mathbb{WE}$ to obtain the following properties of $\mathbb{WE}^*$:

$$
\begin{gathered}
[X \Rightarrow \mathbb{WE}^*.X],\\
\mathit{all\text{-}c}.X \;\equiv\; [\mathbb{WE}^*.X \equiv X],\\
[\mathbb{WE}^*.(X \vee Y) \;\equiv\; \mathbb{WE}^*.X \vee \mathbb{WE}^*.Y],\\
[X \Rightarrow Y] \Rightarrow [\mathbb{WE}^*.X \Rightarrow \mathbb{WE}^*.Y],\\
[X \equiv \mathit{false}] \equiv [\mathbb{WE}^*.X \equiv \mathit{false}] \;\; .
\end{gathered}
$$

5.3 The Property Transformer $\mathbb{SE}^*$

In the same way as what is done in the previous section, we can study the conjugate of the property transformer $\mathbb{SE}$. Applying duality, $\mathbb{SE}^*.X$ is the weakest all-components property stronger than X. From proposition 4, we obtain:

Proposition 8 $\mathbb{SE}^*.X$ *is the weakest property that must be proved on a system to ensure that all components of the system satisfy* X*:*

$$\langle \forall F, G : F \lhd G : Y.G \Rightarrow X.F \rangle \;\equiv\; [Y \Rightarrow \mathbb{SE}^*.X] \;\; .$$

5.4 The Property Transformer $\mathbb{SU}^*$

$\mathbb{SU}^*.X$, as the conjugate of $\mathbb{SU}.X$, is the weakest some-component property stronger than X. By duality of proposition 5:

Proposition 9 $\mathbb{SU}^*.X$ *is the weakest property that must be proved on a system to ensure that at least one component of the system satisfies* X*:*

$$
\begin{aligned}
\langle \forall F, G, H, \cdots : F\surd G \wedge F{\circ}G\surd H \wedge F{\circ}G{\circ}H\surd \cdots :\\
Y \;.\; F{\circ}G{\circ}H{\circ}\cdots \Rightarrow X.F \vee X.G \vee X.H \vee \cdots \rangle\\
\equiv [Y \Rightarrow \mathbb{SU}^*.X] \;\; .
\end{aligned}
$$

5.5 Comparison of the Six Transformers

The following table summarizes the intuitive interpretation for each one of the six property transformers that we have defined. Each transformer corresponds to a different view of composition:

$\mathrm{WE}.X$	What must be proved on a component to ensure that any system that contains that component satisfies X.
$\mathrm{SE}.X$	What can be deduced of any system that contains at least one component that satisfies X.
$\mathrm{SU}.X$	What can be deduced of any system that contains only components that satisfy X.
$\mathrm{WE}^*.X$	What can be deduced on all components of any system that satisfies X.
$\mathrm{SE}^*.X$	What must be proved on a system to ensure that all components satisfy X.
$\mathrm{SU}^*.X$	What must be proved on a system to ensure that at least one component satisfies X.

5.6 Example

We consider another example that uses the bags of balls model: If a system contains exactly one ball and that ball is blue, what can we tell from its components? Using proposition 7, we can claim that all components in such a system must satisfy:

$$\mathrm{WE}^*.(\text{exactly one ball and the ball is blue}) \quad . \tag{12}$$

Applying duality to proposition 6, we deduce:

$$[X \equiv UNIT_=] \vee [\mathrm{WE}^*.(UNIT_{\neq} \wedge X) \equiv \mathrm{WE}^*.X]^{\circledast} \quad .$$

We can then calculate and equivalent formulation for (12):

$$\begin{aligned}
& \mathrm{WE}^*.(\text{exactly one ball and the ball is blue}) \\
= \; & \{\text{Predicate calculus}\} \\
& \mathrm{WE}^*.(UNIT_{\neq} \wedge (\text{all balls are blue}) \wedge (\text{at most one ball})) \\
= \; & \{\text{Apply the dual of prop. 6}\} \\
& \mathrm{WE}^*.((\text{all balls are blue}) \wedge (\text{at most one ball})) \\
= \; & \left\{\begin{array}{l}\text{Both properties are all-components and a conjunction of all-components} \\ \text{properties is an all-components property, for which } [\mathrm{WE}^*.X \equiv X]\end{array}\right\} \\
& (\text{all balls are blue}) \wedge (\text{at most one ball}) \quad .
\end{aligned}$$

This is consistent with our intuition: If a system contains only one ball and the ball is blue, then all its components must contain only blue balls and at most one ball. Note that the condition we obtain is no guaranteed to hold in a system even if it holds in all its components because the property "at most one ball" is not universal.

6 Conclusions

This paper reports on an ongoing exploration of using ideas from the mathematics of program construction to develop a theory of compositional design. This paper shows how we exploit concepts from the axiomatic semantics of programs for designing systems. The specific constructs that we investigated in this paper are predicate transformers and their conjugates. The only assumptions we made about the composition operator were that it was associative and that it had a unit element. A sizable body of theory about transformers can be developed from only these limited assumptions.

We started this study because of our conviction of the importance of composition in systems design. We believe that systems, and especially software systems, should more and more be constructed from generic, "off the shelf", components. This means that *reuse* of systems and components is going to be a central issue.

Reuse of existing systems and description of generic components require a specification language that is abstract enough to be able to specify only the relevant aspects of a system and to hide operational details as much as possible. For this reason, we depart from the process calculus approach, such as in CSP or CCS, and focus on logical specifications. Especially, we are interested in applying our approach to concurrent systems specified with temporal logics.

Of course, more abstract specifications lead to more difficulties in terms of composition. However, a great amount of work has been done in relation with the composition of concurrent systems described in terms of temporal logic specifications. Among the logics that were considered, we can cite the linear temporal logic [15] and some variants [14], TLA [2] or UNITY [9–12, 16]. It should be noted that all these works, and even more general studies done at a semantic [1, 4] or syntactic [3] level, rely on the same fundamental hypothesis that systems are described in terms of states and computations. More precisely, the key point that allows systems to be composed is always the same: Components are specified in terms of *open computations*, i.e., infinite traces that are shared with the environment.

In this paper, we adopt a dual perspective on the question of composition. We do not want to consider systems that can always be composed (i.e., that are specified in such a way that composition is going to work) and we do not rely on the open computations assumption. Instead, we consider *any* kind of systems and look at what happens when they are composed. In this way, we hope to be able to reason on composition in the abstract and understand its fundamental issues.

So far, this proposed framework helped us better understand the *guarantees* operator, which was originally defined in [5], along with existential and universal properties. It is interesting to note that *guarantees*, when applied to temporal logic and reactive systems, gives us a powerful way to combine logical specifications. Especially, the compositional characteristics (existential) of X *guarantees* Y does not depend on properties X and Y. For instance, both the left-hand side and the right-hand side can be progress properties, which is not possible with usual assumption-commitment specifications. This leads to simpler proofs of composition, as in [7].

Also, the framework emphasizes a symmetry between top-down design and bottom-up design. Conjugates of predicate transformers allow us to deduce theorems about top-down decomposition from corresponding theorems of bottom-up composition. We believe that, from a practical point of view, the top-down problem (identify suitable components) is as important as the more classical bottom-up problem (deduce system properties).

In this paper we limited our discussion to properties that were existential or universal or conjunctions of such properties. We have also shown elsewhere [6, 7] that nice proofs of correctness for significant concurrent programs can be developed using these concepts coupled with a logic such as Unity. In this paper we were not concerned with showing how these concepts could be used for programming; instead, we were primarily concerned with showing how concepts from programming can be applied to broad classes of systems in which composition has simple properties such as associativity and existence of a unit element.

Much further work needs to be done to develop an axiomatic semantics of a theory of compositional design. We have only begun to explore the area. Theorems that derive from further assumptions about the compositional operator must be developed. Properties that are not necessarily predicates on systems should be studied. In general, a property is a function from systems to some type that is not limited to booleans. A theory should be able to reason about properties such as mass and energy consumed. We believe that it is possible to construct axiomatic theories that help in understanding the basic principles of compositional design.

References

1. Martín Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, January 1993.
2. Martín Abadi and Leslie Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, May 1995.
3. Martín Abadi and Stephan Merz. An abstract account of composition. In Jivrí Wiedermann and Petr Hajek, editors, *Mathematical Foundations of Computer Science*, volume 969 of *Lecture Notes in Computer Science*, pages 499–508. Springer-Verlag, September 1995.
4. Martín Abadi and Gordon Plotkin. A logical view of composition. *Theoretical Computer Science*, 114(1):3–30, June 1993.
5. K. Mani Chandy and Beverly Sanders. Reasoning about program composition. Submitted for publication.
http://www.cise.ufl.edu/~sanders/pubs/composition.ps.
6. Michel Charpentier and K. Mani Chandy. Examples of program composition illustrating the use of universal properties. In J. Rolim, editor, *International workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA'99)*, volume 1586 of *Lecture Notes in Computer Science*, pages 1215–1227. Springer-Verlag, April 1999.

7. Michel Charpentier and K. Mani Chandy. Towards a compositional approach to the design and verification of distributed systems. In J. Wing, J. Woodcock, and J. Davies, editors, *World Congress on Formal Methods in the Development of Computing Systems (FM'99), (Vol. I)*, volume 1708 of *Lecture Notes in Computer Science*, pages 570–589. Springer-Verlag, September 1999.
8. Michel Charpentier and K. Mani Chandy. Theorems about composition. Technical Report CS-TR-99-02, California Institute of Technology, January 2000. 29 pages.
9. Pierre Collette. *Design of Compositional Proof Systems Based on Assumption-Commitment Specifications. Application to* UNITY. Doctoral thesis, Faculté des Sciences Appliquées, Université Catholique de Louvain, June 1994.
10. Pierre Collette. An explanatory presentation of composition rules for assumption-commitment specifications. *Information Processing Letters*, 50:31–35, 1994.
11. Pierre Collette and Edgar Knapp. Logical foundations for compositional verification and development of concurrent programs in UNITY. In *International Conference on Algebraic Methodology and Software Technology*, volume 936 of *Lecture Notes in Computer Science*, pages 353–367. Springer-Verlag, 1995.
12. Pierre Collette and Edgar Knapp. A foundation for modular reasoning about safety and progress properties of state-based concurrent programs. *Theoretical Computer Science*, 183:253–279, 1997.
13. Edsger W. Dijkstra and Carel S. Scholten. *Predicate calculus and program semantics.* Texts and monographs in computer science. Springer-Verlag, 1990.
14. J.L. Fiadeiro and T. Maibaum. Verifying for reuse: foundations of object-oriented system verification. In I. Makie C. Hankin and R. Nagarajan, editors, *Theory and Formal Methods*, pages 235–257. World Scientific Publishing Company, 1995.
15. Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.
16. Rob T. Udink. *Program Refinement in* UNITY*-like Environments.* PhD thesis, Utrecht University, September 1995.