# Composition Operators in Language Equations

Nina Yevtushenko[¶] Tiziano Villa[§,†] Robert K. Brayton[‡] Alan Mishchenko[‡]
Alberto L. Sangiovanni-Vincentelli[†,‡]

[¶]Dept. of EECS
Tomsk State University
Tomsk, 634050, Russia

[§]DIEGM
Univ. of Udine
33100 Udine, Italy

[†]PARADES
Via di S.Pantaleo, 66
00186 Roma, Italy

[‡]Dept. of EECS
Univ. of California
Berkeley, CA 94720

April 28, 2004

## Abstract

The paper addresses the problem of solving equations over languages, that model the problem of synthesizing an unknown component, both in hardware and software systems. We investigate what algebraic properties must be satisfied by a composition operator so that the related language equation yields a general solution of the same form, and show sufficient conditions to derive such a largest solution from which the solutions of synchronous and parallel equations follow as special cases.

The second contribution is a transformation to study how different composition operators relate to each other, as instantiations of generic templates of abstract language substitution operators, which yield synchronous and parallel composition as special cases.

## 1 Introduction

The paper addresses the problem of solving equations over languages, that model the problem of synthesizing an unknown component, both in hardware and software systems. In [7, 8] we solved the problem for equations defined with respect to synchronous composition (modeling hardware ) and parallel composition (modeling software interleaving). and described the most general solution of those equations in a closed form involving complementation and composition. Here we investigate what algebraic properties must be satisfied by a composition operator so that the related language equation yields a general solution of the same form, and show sufficient conditions to derive such a largest solution from which the solutions of synchronous and parallel equations follow as special cases.

The second contribution is a transformation to study how different composition operators relate to each other, by showing that they can be obtained by instantiating generic templates of abstract language substitution operators, from which we derive synchronous and parallel composition as special cases.

Basic definitions about languages and operators are provided in Sec. 2. Equations over languages are discussed in Sec. 3, where we highlight the sufficient conditions on composition operators that allow to write the solution in a given closed form. The relations between composition operators are discussed in Sec. 4.

## 2 Languages and Operators

### 2.1 Background

**Definition 2.1** *An alphabet is a finite set of symbols. The set of all finite strings over a fixed alphabet $X$ is denoted by $X^\star$. $X^\star$ includes the empty string $\epsilon$. A subset $L \subseteq X^\star$ is called a* **language** *over alphabet $X$.*

Some standard operations on languages are:

1. Given languages $L_1$ and $L_2$, respectively over alphabets $X_1$ and $X_2$, the language $L_1 \cup L_2$ over alphabet $X_1 \cup X_2$ is the **union** of languages $L_1$ and $L_2$.

2. Given languages $L_1$ and $L_2$, respectively over alphabets $X_1$ and $X_2$, the language $L_1L_2 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$ over alphabet $X_1 \cup X_2$ is the **concatenation** of languages $L_1$ and $L_2$. Define $L^0 = \{\epsilon\}$, $L^i = LL^{i-1}$. The **Kleene closure** of $L$ is the set

$L^\star = \cup_{i=0}^{\infty} L^i$ and the **positive Kleene closure** of $L$ is $L^+ = \cup_{i=1}^{\infty} L^i$.

3. Given languages $L_1$ and $L_2$, respectively over alphabets $X_1$ and $X_2$, the language $L_1 \cap L_2$ over alphabet $X_1 \cap X_2$ is the **intersection** of languages $L_1$ and $L_2$. If $X_1 \cap X_2 = \emptyset$ then $L_1 \cap L_2 = \emptyset$.

4. Given a language $L$ over alphabet $X$, the language $\overline{L} = X^\star \setminus L$ over alphabet $X$ is the **complement** of language $L$. Similarly, given languages $L_1$ and $L_2$, respectively over alphabets $X_1$ and $X_2$, the language $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$ over alphabet $X_1$ is the **difference** of languages $L_1$ and $L_2$.

5. Given a language $L$ over alphabet $X$, the language of all prefixes of words in $L$ is *Init*$(L) = \{x \in X^\star \mid \exists y \in X^\star, xy \in L\}$.

It is useful to recall the notions of substitution and homomorphism of languages [3]. A **substitution** $f$ is a mapping of an alphabet $\Sigma$ onto subsets of $\Delta^\star$ for some alphabet $\Delta$. The substitution $f$ is extended to strings by setting $f(\epsilon) = \{\epsilon\}$ and $f(xa) = f(x)f(a)$. An **homomorphism** $h$ is a substitution such that $h(a)$ is a single string for each symbol $a$ in the alphabet $\Sigma$. We introduce some useful operations on languages.

1. Given a language $L$ over alphabet $X \times V$, consider the homomorphism $p : X \times V \to V$ defined as

$$p((x, v)) = v,$$

then the language

$$L_{\downarrow V} = \{p(\alpha) \mid \alpha \in L\}$$

over alphabet $V$ is the **projection** of language $L$ to alphabet $V$, or $V$-projection of $L$. By definition of substitution $p(\epsilon) = \epsilon$.

2. Given a language $L$ over alphabet $X$ and an alphabet $V$, consider the substitution $l : X \to X \times V$ defined as

$$l(x) = \{(x, v) \mid v \in V\},$$

then the language

$$L_{\uparrow V} = \{l(\alpha) \mid \alpha \in L\}$$

over alphabet $X \times V$ is the **lifting** of language $L$ to alphabet $V$, or $V$-lifting of $L$. By definition of substitution $l(\epsilon) = \{\epsilon\}$.

3. Given a language $L$ over alphabet $X \cup V$, consider the homomorphism $r : X \cup V \to V$ defined as

$$r(y) = \begin{cases} y & \text{if } y \in V \\ \epsilon & \text{if } y \in X \setminus V \end{cases},$$

then the language

$$L_{\Downarrow V} = \{r(\alpha) \mid \alpha \in L\}$$

over alphabet $V$ is the **restriction** of language $L$ to alphabet $V$, or $V$-restriction of $L$, i.e., words in $L_{\Downarrow V}$ are obtained from those in $L$ by deleting all the symbols in $X$ that are not symbols in $V$. By definition of substitution $r(\epsilon) = \epsilon$.

4. Given a language $L$ over alphabet $X$ and an alphabet $V$ disjoint from $X$, consider the mapping $e : X \to X \cup V$ defined as

$$e(x) = \{\alpha x \beta \mid \alpha, \beta \in V^\star\},$$

then the language

$$L_{\Uparrow V} = \{e(\alpha) \mid \alpha \in L\}$$

over alphabet $X \cup V$ is the **expansion** of language $L$ to alphabet $V$, or $V$-expansion of $L$, i.e., words in $L_{\Uparrow V}$ are obtained from those in $L$ by inserting anywhere in them words from $V^\star$. Notice that $e$ is not a substitution and that $e(\epsilon) = \{\alpha \mid \alpha \in V^\star\}$.

By definition $\emptyset_{\downarrow V} = \emptyset$, $\emptyset_{\uparrow V} = \emptyset$, $\emptyset_{\Downarrow V} = \emptyset$, $\emptyset_{\Uparrow V} = \emptyset$.

## 2.2 Algebraic Properties of Operators

The following straightforward facts hold between the projection and lifting operators, and between the restriction and expansion operators. In the following, unless otherwise stated, the union is taken over non-disjoint alphabets.

**Proposition 2.1** *The following relations hold.*
*(a) Given alphabets $X$ and $Y$, and a language $L$ over alphabet $X$, then $(L_{\uparrow Y})_{\downarrow X} = L$.*
*(b) Given alphabets $X$ and $Y$, and a language $L$ over alphabet $X \times Y$, then $(L_{\downarrow X})_{\uparrow Y} \supseteq L$.*
*(c) Given alphabets $X$ and $Y$ ($X$, $Y$ disjoint), and a language $L$ over alphabet $X$, then $(L_{\Uparrow Y})_{\Downarrow X} = L$.*
*(d) Given alphabets $X$ and $Y$ ($X$, $Y$ disjoint), and a language $L$ over alphabet $X \cup Y$, then $(L_{\Downarrow X})_{\Uparrow Y} \supseteq L$.*

**Proposition 2.2** *The following distributive laws for $\uparrow$ and $\downarrow$ hold.*
*(a) Let $L_1, L_2$ be languages over alphabet $U$. Then $\uparrow$ commutes with $\cup$*

$$(L_1 \cup L_2)_{\uparrow I} = L_{1\uparrow I} \cup L_{2\uparrow I}.$$

*(b) Let $L_1, L_2$ be languages over alphabet $U$. Then $\uparrow$ commutes with $\cap$*

$$(L_1 \cap L_2)_{\uparrow I} = L_{1\uparrow I} \cap L_{2\uparrow I}.$$

*(c) Let $M_1, M_2$ be languages over alphabet $I \times U$. Then $\downarrow$ commutes with $\cup$*

$$(M_1 \cup M_2)_{\downarrow U} = M_{1\downarrow U} \cup M_{2\downarrow U}.$$

*(d) Let $M_1, M_2$ be languages over alphabet $I \times U$. If $M_2 = (M_2 \downarrow_U)\uparrow_I$ (or $M_1 = (M_1 \downarrow_U)\uparrow_I$) then $\downarrow$ commutes with $\cap$*

$$(M_1 \cap M_2)\downarrow_U = M_1 \downarrow_U \cap M_2 \downarrow_U.$$

**Proof.** $(L_1 \cap L_2)\uparrow_I = L_1 \uparrow_I \cap L_2 \uparrow_I$.

($\Rightarrow$) If the string $(i_1, u_1) \dots (i_k, u_k) \in (L_1 \cap L_2)\uparrow_I$, then $u_1 \dots u_k \in L_1 \cap L_2$; thus $u_1 \dots u_k \in L_1$, $u_1 \dots u_k \in L_2$, and so $(i_1, u_1) \dots (i_k, u_k) \in L_1 \uparrow_I$, $(i_1, u_1) \dots (i_k, u_k) \in L_2 \uparrow_I$, implying $(i_1, u_1) \dots (i_k, u_k) \in L_1 \uparrow_I \cap L_2 \uparrow_I$.

($\Leftarrow$) If the string $(i_1, u_1) \dots (i_k, u_k) \in L_1 \uparrow_I \cap L_2 \uparrow_I$, then $(i_1, u_1) \dots (i_k, u_k) \in L_1 \uparrow_I$, $(i_1, u_1) \dots (i_k, u_k) \in L_2 \uparrow_I$; thus $u_1 \dots u_k \in L_1$, $u_1 \dots u_k \in L_2$, implying $u_1 \dots u_k \in L_1 \cap L_2$, and so $(i_1, u_1) \dots (i_k, u_k) \in (L_1 \cap L_2)\uparrow_I$.

Similarly one proves the first and third identity involving $\cup$.

$(M_1 \cap M_2)\downarrow_U = M_1 \downarrow_U \cap M_2 \downarrow_U$.

($\Rightarrow$) If the string $u_1 \dots u_k \in (M_1 \cap M_2)\downarrow_U$ then there exists $i_1 \dots i_k$ such that $(i_1, u_1) \dots (i_k, u_k) \in M_1 \cap M_2$, i.e., $(i_1, u_1) \dots (i_k, u_k) \in M_1$, $(i_1, u_1) \dots (i_k, u_k) \in M_2$, and so $u_1 \dots u_k \in M_1 \downarrow_U$ and $u_1 \dots u_k \in M_2 \downarrow_U$.

($\Leftarrow$) If the string $u_1 \dots u_k \in M_1 \downarrow_U \cap M_2 \downarrow_U$, i.e., $u_1 \dots u_k \in M_1 \downarrow_U$ and $u_1 \dots u_k \in M_2 \downarrow_U$, then there exists $i_1 \dots i_k$ such that $(i_1, u_1) \dots (i_k, u_k) \in M_1$. Moreover, since $M_2 = (M_2 \downarrow_U)\uparrow_I$, from $u_1 \dots u_k \in M_2 \downarrow_U$ it follows that $(i_1, u_1) \dots (i_k, u_k) \in M_2$. In summary, $(i_1, u_1) \dots (i_k, u_k) \in M_1$ and $(i_1, u_1) \dots (i_k, u_k) \in M_2$, implying $(i_1, u_1) \dots (i_k, u_k) \in M_1 \cap M_2$, from which follows $u_1 \dots u_k \in (M_1 \cap M_2)\downarrow_U$. □

**Example 2.1** *We show that the last identity $(M_1 \cap M_2)\downarrow_U = M_1 \downarrow_U \cap M_2 \downarrow_U$ does not hold without an additional hypothesis. Consider $I = \{a, b\}$, $U = \{u\}$, $M_1 = \{au\}$, $M_2 = \{bu\}$, then $(M_1 \cap M_2)\downarrow_U = \emptyset\downarrow_U = \emptyset$ and $M_1 \downarrow_U \cap M_2 \downarrow_U = \{u\} \cap \{u\} = \{u\}$. Notice that $au$ and $bu$ are words of length 1 on the alphabet $I \times U$.*

**Proposition 2.3** *The following distributive laws for $\Uparrow$ and $\Downarrow$ hold.*
*(a) Let $L_1, L_2$ be languages over alphabet $U$. Then $\Uparrow$ commutes with $\cup$*

$$(L_1 \cup L_2)\Uparrow_I = L_1 \Uparrow_I \cup L_2 \Uparrow_I.$$

*(b) Let $L_1, L_2$ be languages over alphabet $U$. Then $\Uparrow$ commutes with $\cap$*

$$(L_1 \cap L_2)\Uparrow_I = L_1 \Uparrow_I \cap L_2 \Uparrow_I.$$

*(c) Let $M_1, M_2$ be languages over alphabet $I \cup U$. Then $\Downarrow$ commutes with $\cup$*

$$(M_1 \cup M_2)\Downarrow_U = M_1 \Downarrow_U \cup M_2 \Downarrow_U.$$

*(d) Let $M_1, M_2$ be languages over alphabet $I \cup U$. If $M_2 = (M_2 \Downarrow_U)\Uparrow_I$ (or $M_1 = (M_1 \Downarrow_U)\Uparrow_I$) then $\Downarrow$ commutes with $\cap$*

$$(M_1 \cap M_2)\Downarrow_U = M_1 \Downarrow_U \cap M_2 \Downarrow_U.$$

**Proof.** $(L_1 \cap L_2)\Uparrow_I = L_1 \Uparrow_I \cap L_2 \Uparrow_I$.

($\Rightarrow$) If the string $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in (L_1 \cap L_2)\Uparrow_I$ and $\alpha_1, \dots, \alpha_k, \alpha_{k+1} \in I^\star$, then $u_1 \dots u_k \in L_1 \cap L_2$; thus $u_1 \dots u_k \in L_1$, $u_1 \dots u_k \in L_2$, and so $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_1 \Uparrow_I$, $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_2 \Uparrow_I$, implying $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_1 \Uparrow_I \cap L_2 \Uparrow_I$.

($\Leftarrow$) If the string $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_1 \Uparrow_I \cap L_2 \Uparrow_I$, then it holds that $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_1 \Uparrow_I$ and $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in L_2 \Uparrow_I$; thus $u_1 \dots u_k \in L_1$, $u_1 \dots u_k \in L_2$, implying that $u_1 \dots u_k \in L_1 \cap L_2$, and so $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in (L_1 \cap L_2)\Uparrow_I$.

Similarly one proves the first and third identity involving $\cup$.

$(M_1 \cap M_2)\Downarrow_U = M_1 \Downarrow_U \cap M_2 \Downarrow_U$.

($\Rightarrow$) If the string $u_1 \dots u_k \in (M_1 \cap M_2)\Downarrow_U$ then there are $\alpha_1, \dots \alpha_k, \alpha_{k+1} \in I^\star$ such that $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_1 \cap M_2$, i.e., $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_1$, $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_2$, and so $u_1 \dots u_k \in M_1 \Downarrow_U$ and $u_1 \dots u_k \in M_2 \Downarrow_U$.

($\Leftarrow$) If the string $u_1 \dots u_k \in M_1 \Downarrow_U \cap M_2 \Downarrow_U$, i.e., $u_1 \dots u_k \in M_1 \Downarrow_U$ and $u_1 \dots u_k \in M_2 \Downarrow_U$, then there exists $\alpha_1 \dots \alpha_k \alpha_{k+1} \in I^\star$ such that $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_1$. Moreover, since $M_2 = (M_2 \Downarrow_U)\Uparrow_I$, from $u_1 \dots u_k \in M_2 \Downarrow_U$ it follows that $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_2$. In summary, $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_1$ and $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_2$, implying $\alpha_1 u_1 \dots \alpha_k u_k \alpha_{k+1} \in M_1 \cap M_2$, from which follows $u_1 \dots u_k \in (M_1 \cap M_2)\Downarrow_U$. □

**Example 2.2** *We show that the last identity $(M_1 \cap M_2)\Downarrow_U = M_1 \Downarrow_U \cap M_2 \Downarrow_U$ does not hold without an additional hypothesis. Consider $I = \{a, b\}$, $U = \{u\}$, $M_1 = \{au\}$, $M_2 = \{bu\}$, then $(M_1 \cap M_2)\Downarrow_U = \emptyset\Downarrow_U = \emptyset$ and $M_1 \Downarrow_U \cap M_2 \Downarrow_U = \{u\} \cap \{u\} = \{u\}$. Notice that $au$ and $bu$ are words of length 2 on the alphabet $I \cup U$.*

Regular languages, corresponding to finite automata, play a special role in hardware and software synthesis. Regular languages are closed under union, concatenation, complementation and intersection. Also regular languages are closed under projection, lifting and restriction, because they are closed under substitution [3]. Regular languages are closed under expansion, because there is an algorithm that, given the finite automaton of a language, returns the finite automaton of the expanded language.

## 2.3 Composition of Languages

Consider two systems $A$ and $B$ with associated languages $L(A)$ and $L(B)$. The systems communicate with each

other by a channel $U$ and with the environment by channels $I$ and $O$. We introduce two composition operators that describe the external behaviour of the composition of $L(A)$ and $L(B)$.

**Definition 2.2** *Given the disjoint alphabets $I, U, O$, language $L_1$ over $I \times U$ and language $L_2$ over $U \times O$, the* **synchronous composition** *of languages $L_1$ and $L_2$ is the language* [1] $[(L_1)_{\uparrow O} \cap (L_2)_{\uparrow I}]_{\downarrow I \times O}$*, denoted by $L_1 \bullet L_2$, defined over $I \times O$.*

**Definition 2.3** *Given the disjoint alphabets $I, U, O$, language $L_1$ over $I \cup U$ and language $L_2$ over $U \cup O$, the* **parallel composition** *of languages $L_1$ and $L_2$ is the language $[(L_1)_{\Uparrow O} \cap (L_2)_{\Uparrow I}]_{\Downarrow I \cup O}$, denoted by $L_1 \diamond L_2$, defined over $I \cup O$.*

By definition of the operations $\downarrow_V$, $\uparrow_V$, $\Downarrow_V$, $\Uparrow_V$, $\Uparrow_{(V,l)}$ it follows that $\emptyset \bullet L = L \bullet \emptyset = \emptyset$, $\emptyset \diamond L = L \diamond \emptyset = \emptyset$.

Variants of *synchronous composition* are introduced in [2] as *product*, $\times$ (with the comment *sometimes called completely synchronous composition*), and in [4] as *synchronous parallel composition*, $\otimes$. Variants of *parallel composition* are introduced in [2] as *parallel composition*, $\|$ (with the comment *often called synchronous composition*), and in [4] as *interleaving parallel composition*, $\|$; the same operator was called *asynchronous composition* in [6]. These definitions were usually introduced for regular languages; actually they were more commonly given for finite automata.

It has also been noticed by Kurshan [4] and Arnold [1] that asynchronous systems can also be modeled with the synchronous interpretation, using null transitions to keep a transition system in the same state for an arbitrary period of time. Kurshan [4] observes that: "While synchronous product often is thought to be a simple -even uninteresting!- type of coordination, it can be shown that, through use of nondeterminism, this conceptually simple coordination serves to model the most general 'asynchronous' coordination, i.e., where processes progress at arbitrary rates relative to one another. In fact the 'interleaving' model, the most common model for asynchrony in the software community, can be viewed as a special case of this synchronous product." A discussion can be found in [5].

# 3 Equations over Languages

## 3.1 Language Equations under Abstract Composition

In this section we introduce symbols to represent operators defined over alphabets and languages, namely:

- let $\circ$ denote an operator between alphabets, e.g., $\circ$ could be $\times$ or $\cup$;
- let $\odot$ denote a composition operator between languages, defined as

$$L_1 \odot L_2 = [(L_1)_{\top O} \cap (L_2)_{\top I}]_{\bot I \circ O}$$

where $\top$ and $\bot$ denote a language substitution operator, e.g., $\top$ could be $\uparrow$, $\bot$ could be $\downarrow$, and so $\odot$ would denote the synchronous composition operator $\bullet$ and $\circ$ would denote $\times$ (or $\top$ could be $\Uparrow$, $\bot$ could be $\Downarrow$, and so $\odot$ would denote the parallel composition operator $\diamond$ and $\circ$ would denote $\cup$).

Given the disjoint alphabets $I, U, O$, a language $A$ over alphabet $I \circ U$ and a language $C$ over alphabet $I \circ O$, consider the language equation

$$A \odot X \subseteq C. \tag{1}$$

We address the question of providing a closed-form solution of such language equation with respect to an abstract composition operator $\odot$ under the less restrictive algebraic conditions.

**Definition 3.1** *Given alphabets $I, U, O$, a language $A$ over alphabet $I \circ U$ and a language $C$ over alphabet $I \circ O$, language $B$ over alphabet $U \circ O$ is called a* **solution** *of the equation $A \odot X \subseteq C$ iff $A \odot B \subseteq C$. A solution is called the* **largest solution** *if it contains any other solution. $B = \emptyset$ is the trivial solution.*

The following theorem states requirements on the substitution operators $\top$ and $\bot$ that allow writing the largest solution in the closed-form $S = \overline{A \odot \overline{C}}$.

**Theorem 3.1** *If the substitution operators $\top$ and $\bot$ are such that*

$H1$ : *Given alphabets $X, Y$ and language $L$ over $X$, $(L_{\top Y})_{\bot X} = L$*

$H2$ : *Given alphabets $X, Y$ and languages $L_1, L_2$ over $Y \circ X$, if $L_1 = (L_{1\bot X})_{\top Y}$ or $L_2 = (L_{2\bot X})_{\top Y}$ then $(L_1 \cap L_2)_{\bot X} = L_{1\ \bot X} \cap L_{2\ \bot X}$*

*then the largest solution of the equation $A \odot X \subseteq C$ is the language $S = \overline{A \odot \overline{C}}$.*

**Proof.** Consider a string $\alpha \in (U \circ O)^\star$, then $\alpha$ is in the largest solution of $A \odot X \subseteq C$ iff $A \odot \{\alpha\} \subseteq C$ and the

[1] Use the same order $I \times U \times O$ in the languages $(L_1)_{\uparrow O}$ and $(L_2)_{\uparrow I}$.

following chain of equivalences follows:

$$\begin{aligned}
A \odot \{\alpha\} \subseteq C &\quad \Leftrightarrow \\
(A_{\top O} \cap \{\alpha\}_{\top I})_{\perp I\circ O} \cap \overline{C} = \emptyset &\quad \Leftrightarrow \\
\textit{by Hyp. } H1: \ \overline{C} = (\overline{C}_{\top U})_{\perp I\circ O} & \\
(A_{\top O} \cap \{\alpha\}_{\top I})_{\perp I\circ O} \cap (\overline{C}_{\top U})_{\perp I\circ O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Hyp. } H2: \ \cap \textit{ and } \perp \textit{ commute} & \\
(A_{\top O} \cap \{\alpha\}_{\top I} \cap \overline{C}_{\top U})_{\perp I\circ O} = \emptyset &\quad \Leftrightarrow \\
A_{\top O} \cap \{\alpha\}_{\top I} \cap \overline{C}_{\top U} = \emptyset &\quad \Leftrightarrow \\
(A_{\top O} \cap \{\alpha\}_{\top I} \cap \overline{C}_{\top U})_{\perp U\circ O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Hyp. } H2: \ \cap \textit{ and } \perp \textit{ commute} & \\
(\{\alpha\}_{\top I})_{\perp U\circ O} \cap (A_{\top O} \cap \overline{C}_{\top U})_{\perp U\circ O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Hyp. } H1: \ (\{\alpha\}_{\top I})_{\perp U\circ O} = \{\alpha\} & \\
\{\alpha\} \cap (A_{\top O} \cap \overline{C}_{\top U})_{\perp U\circ O} = \emptyset &\quad \Leftrightarrow \\
\alpha \notin (A_{\top O} \cap \overline{C}_{\top U})_{\perp U\circ O} &\quad \Leftrightarrow \\
\alpha \in \overline{(A_{\top O} \cap \overline{C}_{\top U})_{\perp U\circ O}} &\quad \Leftrightarrow \\
\alpha \in \overline{A \odot \overline{C}} &
\end{aligned}$$

Therefore the largest solution of the language equation $A \odot X \subseteq C$ is given by the language

$$S = \overline{A \odot \overline{C}}. \tag{2}$$

□

It is an open question whether the previous sufficient conditions are the less restrictive ones and so also necessary.

## 3.2 Language Equations under Synchronous and Parallel Composition

We can deduce as corollaries of Th. 3.1 the solutions of synchronous equation $A \bullet X \subseteq C$ and of parallel equation $A \diamond X \subseteq C$.

Given the disjoint alphabets $I, U, O$, a language $A$ over alphabet $I \times U$ and a language $C$ over alphabet $I \times O$, consider the language equation

$$A \bullet X \subseteq C. \tag{3}$$

**Definition 3.2** *Given alphabets $I, U, O$, a language $A$ over alphabet $I \times U$ and a language $C$ over alphabet $I \times O$, language $B$ over alphabet $U \times O$ is called a* **solution** *of the equation $A \bullet X \subseteq C$ iff $A \bullet B \subseteq C$. A solution is called the* **largest solution** *if it contains any other solution. $B = \emptyset$ is the trivial solution.*

**Corollary 3.1** *The largest solution of the equation $A \bullet X \subseteq C$ is the language $S = \overline{A \bullet \overline{C}}$.*

**Proof.** The proof follows from Th. 3.1, noticing that Hyp. H1 holds due to Prop. 2.1(a) and Hyp. H2 holds due to Prop. 2.2(d). The hypothesis under which the latter proposition holds is verified (in the first instance because $((\overline{C}_{\uparrow U})_{\downarrow I\times O})_{\uparrow U} = \overline{C}_{\uparrow U}$ and in the second instance because $\{\alpha\}_{\uparrow I} = ((\{\alpha\}_{\uparrow I})_{\downarrow U\times O})_{\uparrow I}$).

For ease of reading we repeat the steps of the proof specialized to synchronous composition.

Consider a string $\alpha \in (U \times O)^\star$, then $\alpha$ is in the largest solution of $A \bullet X \subseteq C$ iff $A \bullet \{\alpha\} \subseteq C$ and the following chain of equivalences follows:

$$\begin{aligned}
A \bullet \{\alpha\} \subseteq C &\quad \Leftrightarrow \\
(A_{\uparrow O} \cap \{\alpha\}_{\uparrow I})_{\downarrow I\times O} \cap \overline{C} = \emptyset &\quad \Leftrightarrow \\
\textit{by Prop. } 2.1(a)\ \overline{C} = (\overline{C}_{\uparrow U})_{\downarrow I\times O} & \\
(A_{\uparrow O} \cap \{\alpha\}_{\uparrow I})_{\downarrow I\times O} \cap (\overline{C}_{\uparrow U})_{\downarrow I\times O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Prop. } 2.2(d) \textit{ since } ((\overline{C}_{\uparrow U})_{\downarrow I\times O})_{\uparrow U} = \overline{C}_{\uparrow U} & \\
(A_{\uparrow O} \cap \{\alpha\}_{\uparrow I} \cap \overline{C}_{\uparrow U})_{\downarrow I\times O} = \emptyset &\quad \Leftrightarrow \\
A_{\uparrow O} \cap \{\alpha\}_{\uparrow I} \cap \overline{C}_{\uparrow U} = \emptyset &\quad \Leftrightarrow \\
(A_{\uparrow O} \cap \{\alpha\}_{\uparrow I} \cap \overline{C}_{\uparrow U})_{\downarrow U\times O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Prop. } 2.2(d) \textit{ since } \{\alpha\}_{\uparrow I} = ((\{\alpha\}_{\uparrow I})_{\downarrow U\times O})_{\uparrow I} & \\
(\{\alpha\}_{\uparrow I})_{\downarrow U\times O} \cap (A_{\uparrow O} \cap \overline{C}_{\uparrow U})_{\downarrow U\times O} = \emptyset &\quad \Leftrightarrow \\
\textit{by Prop. } 2.1(a)\ (\{\alpha\}_{\uparrow I})_{\downarrow U\times O} = \{\alpha\} & \\
\{\alpha\} \cap (A_{\uparrow O} \cap \overline{C}_{\uparrow U})_{\downarrow U\times O} = \emptyset &\quad \Leftrightarrow \\
\alpha \notin (A_{\uparrow O} \cap \overline{C}_{\uparrow U})_{\downarrow U\times O} &\quad \Leftrightarrow \\
\alpha \in \overline{(A_{\uparrow O} \cap \overline{C}_{\uparrow U})_{\downarrow U\times O}} &\quad \Leftrightarrow \\
\alpha \in \overline{A \bullet \overline{C}} &
\end{aligned}$$

□

Given pairwise disjoint alphabets $I, U, O$, a language $A$ over alphabet $I \cup U$ and a language $C$ over alphabet $I \cup O$, consider the language equation

$$A \diamond X \subseteq C. \tag{4}$$

**Definition 3.3** *Given pairwise disjoint alphabets $I, U, O$, a language $A$ over alphabet $I \cup U$ and a language $C$ over alphabet $I \cup O$, language $B$ over alphabet $U \cup O$ is called a* **solution** *of the equation $A \diamond X \subseteq C$ iff $A \diamond B \subseteq C$. The* **largest solution** *is a solution that contains any other solution. $B = \emptyset$ is the trivial solution.*

**Corollary 3.2** *The largest solution of the equation $A \diamond X \subseteq C$ is the language $S = \overline{A \diamond \overline{C}}$.*

**Proof.** The proof follows from Th. 3.1, noticing that Hyp. H1 holds due to Prop. 2.1(c) and Hyp. H2 holds due to Prop. 2.3(d). The hypothesis under which the latter proposition holds is verified (in the first instance because $((\overline{C}_{\Uparrow U})_{\Downarrow I\cup O})_{\Uparrow U} = \overline{C}_{\Uparrow U}$ and in the second instance because $\{\alpha\}_{\Uparrow I} = ((\{\alpha\}_{\Uparrow I})_{\Downarrow U\cup O})_{\Uparrow I}$). □

# 4 Relation between Composition Operators

This section introduces a transformation that sheds light on how different composition operators relate to each other, by showing that they can be obtained by instantiating generic templates of $\top$ and $\bot$ language substitution operators. In particular we show how synchronous and parallel composition are obtained by specializing the $\top$ and $\bot$ operators.

We suppose to introduce a new alphabet symbol $\mu$ ("silent" symbol) that is different from any existing alphabet symbol, and we add $\mu$ to every alphabet under consideration.

Given the languages $L_A$ over alphabet $I \cup U$ and $L_B$ over alphabet $U \cup O$, consider the following transformation that yields the languages $\tilde{L}_A$ and $\tilde{L}_B$ over alphabet $I \times U \times O$.
For each string in $L_A$ or in $L_B$:

1. replace each symbol $i$ by the triple $i\mu\mu$, each symbol $u$ by $\mu u\mu$ and each symbol $o$ by $\mu\mu o$;
2. insert the regular expression $(\mu\mu\mu)^\star$ after each word prefix.

**Example 4.1** *The string $i_1u_2u_1i_2 \in L_A$ is transformed into the set of strings*

$$(\mu\mu\mu)^\star i_1\mu\mu(\mu\mu\mu)^\star\mu u_2\mu(\mu\mu\mu)^\star\mu u_1\mu(\mu\mu\mu)^\star i_2\mu\mu(\mu\mu\mu)^\star \in \tilde{L}_A.$$

Similarly, given the languages $L_A$ over alphabet $I \times U$ and $L_B$ over alphabet $U \times O$, consider the following transformation that yields the languages $\tilde{L}_A$ and $\tilde{L}_B$ over alphabet $I \times U \times O$.
For each string in $L_A$ or in $L_B$:

1. replace each pair $iu$ by the triple $iu\mu$ and each pair $uo$ by the triple $\mu uo$;
2. insert the regular expression $(\mu\mu\mu)^\star$ after each word prefix.

The previous transformations can be generalized to languages over alphabets obtained by a composition of $\cup$ and $\times$ operators (where $\cup$ models the interleaving of channels and $\times$ models the synchronization of channels). By deleting the occurrences of the silent symbol $\mu$ one goes back from $\tilde{L}_A$ and $\tilde{L}_B$ to $L_A$ and $L_B$.

This transformation can be used to solve parallel equations by means of a solver for synchronous equations.

Given the transformed languages $\tilde{L}_A$ and $\tilde{L}_B$, each defined over alphabet $I \times U \times O$, introduce now the language substitution operators $\top I : I \times U \times O \to 2^{I\times U\times O}$ and $\top O : I \times U \times O \to 2^{I\times U\times O}$, and the language homomorphism operator $\bot IO : I \times U \times O \to I \times U \times O$.

**Definition 4.1** *$L_A \odot L_B$ is the language obtained by deleting all silent symbols $\mu$ from the language*

$$(\tilde{L}_{A\,\top O} \cap \tilde{L}_{B\,\top I})_{\bot IO}.$$

By choosing specific instances of the $\top$ and $\bot$ operators one gets the well-known synchronous and parallel composition operators (and many more, according to what channels are synchronized and which ones are interleaved).

**Synchronous composition, $\bullet$,** is obtained by the following choices

- $\top O$ For $\top_O$ to realize $\uparrow_O$, each triple $iu\mu$ is replaced by the set $\{iuo \mid o \in O\}$ and the triple $\mu\mu\mu$ is replaced by the set $\{\mu\mu\mu\}$;
- $\top I$ For $\top_I$ to realize $\uparrow_I$, each triple $\mu uo$ is replaced by the set $\{iuo \mid i \in I\}$ and the triple $\mu\mu\mu$ is replaced by the set $\{\mu\mu\mu\}$;
- $\bot IO$ For $\bot_{IO}$ to realize $\downarrow_{I\times O}$, each triple $iuo$ is replaced by the triple $i\mu o$, whereas the triple $\mu\mu\mu$ stays the same.

**Parallel composition, $\diamond$,** is obtained by the following choices

- $\top O$ For $\top_O$ to realize $\Uparrow_O$, the triples $i\mu\mu$ and $\mu u\mu$ stay the same, whereas the triple $\mu\mu\mu$ is replaced by the set $\{\mu\mu o \mid o \in O\}$;
- $\top I$ For $\top_I$ to realize $\Uparrow_I$, the triples $\mu u\mu$ and $\mu\mu o$ stay the same, whereas the triple $\mu\mu\mu$ is replaced by the set $\{i\mu\mu \mid i \in I\}$;
- $\bot IO$ For $\bot_{IO}$ to realize $\Downarrow_{I\times O}$, the triples $i\mu\mu$, $\mu\mu o$ and $\mu\mu\mu$ stay the same, whereas the triple $\mu u\mu$ is replaced by the triple $\mu\mu\mu$.

# 5 Conclusions

This paper reports work in progress on the semantics of composition operators deployed in the definition and solution of the unknown component problem. It states a set of sufficient conditions to be satisfied by a composition operator so that the related language equation admits a given closed-form solution.

Moreover, it argues how different composition operators can be obtained as specific instances of abstract language substitution operators, showing in particular the derivation of synchronous and parallel composition.

# 6 Acknowledgments

The authors gratefully acknowledge the support of a NATO travel grant 2003-2004 (NATO Linkage Grant No. PST.CLG.979698). The first author was partly supported by the scientific program "Russian Universities".

# References


[1] A. Arnold. *Finite transition systems : semantics of communicating systems*. Prentice Hall, 1994.

[2] C. C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.

[3] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 2001.

[4] R.P. Kurshan. *Computer-aided verification of coordinating processes*. Princeton University Press, 1994.

[5] R.P. Kurshan, M. Merritt, A. Orda, and S.R. Sachs. Modelling asynchrony with a synchronous model. *Formal Methods in System Design*, vol. 15(no. 3):175–199, November 1999.

[6] A. Petrenko and N. Yevtushenko. Solving asynchronous equations. In S. Budkowski, A. Cavalli, and E. Najm, editors, *Formal Description Techniques and Protocol Specification, Testing and Verification - FORTE XI/PSTV XVIII '98*, pages 231–247. Kluwer Academic Publishers, November 1998.

[7] N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli. Solution of parallel language equations for logic synthesis. In *The Proceedings of the International Conference on Computer-Aided Design*, pages 103–110, November 2001.

[8] N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli. Solution of synchronous language equations for logic synthesis. In *The Biannual 4th Russian Conference with Foreign Participation on Computer-Aided Technologies in Applied Mathematics*, September 2002.