NONDETERMINISTIC SEMANTICS OF COMPOUND DIAGRAMS

FAIROUZ TCHIER
Mathematics department, King Saud University
P.O.Box 22452, Riyadh 11495, Saudi Arabia

Abstract. We presented a unified description of flow control and single steps of a program is given to obtain flexible definitions of algebraic manipulations. This is achieved by using the notion of relational diagram. We show how the notion of relational diagram, introduced by Schmidt, can be used to give a demonic definition for a wide range of programming constructs. It is shown that the input-output relation of a compound diagram is equal to that of the diagram in which each sub-diagram has been replaced by its input-output relation. This process is repeated until elementary diagrams is obtained.

1. Introduction. Programming languages, semantics describes the behavior that a computer follows when executing a program in the language. It gives the relationship between the input and output of a program. In other words, the semantics of the language will decide which sentences are meaningful and which are nonsense [4, 29, 22]. A basic idea of relational semantics is to let the “meaning” of a program statement be a relation connecting inputs (initial states) with outputs (final states).

The nondeterminism concept has been recognized as a vital component of designing programs and calculating their semantics [4, 6, 9]. Consider the following situation arising in programming. At a certain point further action depends on the choice between a number of alternatives, the method of choice between these alternatives is not directly specified by the programmer; the program must choose between the alternatives. A programmer specifies a limited number of alternatives, but the program must later choose between them (“Choose” is, in fact, a typical name for the nondeterministic programs).

Binary relations are suitable for nondeterministic programs language because a relation has the possibility to have many outputs so it can represent the input and output values of the program.

In the presence of nondeterminism, one can adopt an angelic view, by considering that a program produces a result when there is a possibility to do so or a demonic view, by considering that a program will produce no result when it is possible. During the execution of a program in an input state, the angelic semantics is the input/output relation obtained by considering the best execution of the program if there is a possibility to terminate it will terminate (like an angel it looks for the best). The angelic semantics corresponds to the choice that termination is guaranteed as long as termination is possible. The demonic semantics is the input/output relation by considering the worst execution of the program; if there is a possibility for the

2010 Mathematics Subject Classification. 18C10, 18C50, 68Q55, 68Q65, 03B70, 06B35.

Key words and phrases. Relational compound diagrams, demonic semantics, graphs, monotypes, residuals.
program not to terminate normally, then it will not terminate normally (like a
demon it looks for the worst case). The nondeterminism is important because, for a
computation to be successful all possible executions must lead to a successful result
[2, 3, 6, 9, 46, 47, 48, 49, 50, 51, 52].

Graphs are usually used in computer science to represent programs, automata
and Turing machines [11]. These graphs are composed of vertices connected by orien-
ted arcs. Vertices represent control nodes and arcs represent possible transitions of
control between commands. Graphs have been used by many authors [16, 27, 34, 40],
where they were introduced with different purposes and defined in connection with
the descriptions of algorithms or programs.

In a relational formalism, a graph is based on relational concepts : a representa-
tion relation of the graph, a set of vertices, an initial vertex and a final vertex.
By using the same approach as above, we define a diagram as being a quadruple
constructed of a relation, a set of partial identities disjoint from each other (they
have a role identical to the vertices of a graph) and also of two particular partial
identities characterizing the input and the output of the diagram.

By using these concepts, we show that we can extract from diagrams a binary
input-output relation by considering their worst execution it is the demonic rela-
tional semantics.

We use the same approach as Schmidt and Ströhlein [42]; they defined a re-
lational program as being a quintuple made of a situation graph, a flowgraph, a
relational homomorphism, an input relation and an output relation (for more de-
tails see Chapter 10 in [42]). Schmidt and Ströhlein have used these notions to treat
the total correctness and the partial correctness of programs.

Our approach uses demonic operations. The operators \( \lor \) and \( \circ \) serve to give
an angelic semantics. The demonic operators \( \sqcup \) and \( \Box \) do the opposite, they are
used to give the demonic semantics of nondeterministic programs. In this paper,
we present an approach to analyze the termination properties of these programs.

This paper is organized as follows. In Section 2, we present our mathematical
tool, abstract relation algebra [14, 42, 45]. In Section 3, we define recall some basic
laws and notions related to infinite looping. In Section 4, we describe our refinement
ordering and some demonic operators associated with this order. In Section 5, we
define the diagrams and we give two types of diagrams : elementary and compound
diagrams. In Section 6, we give a formal definition of the demonic input-output
relation of a diagram. This definition encompasses many types of diagrams. In
Section 7, we give an application of our results. We conclude in Section 8 with
prospects for future research.

2. Relation algebras.

2.1. Definition and basic laws.

**Definition 2.1.** A relation algebra \( A \) is a structure \( (B, \lor, \land, \neg, \circ, \tilde{\circ}) \) over a non-
empty set \( B \) of elements, called relations. The unary operations \( \neg, \tilde{\circ} \) are total
whereas the binary operations \( \lor, \land, \circ \) are partial. We denote by \( B_{\lor R} \) the set of
those elements \( Q \in B \) for which the union \( R \lor Q \) is defined and we require that
\( R \in B_{\lor R} \) for every \( R \in B \). If \( Q \in B_{\lor R} \), we say that \( Q \) has the same type as \( R \). The
following conditions are satisfied.

(a) \( (B_{\lor R}, \lor, \land, \neg) \) is a Boolean algebra, with zero element \( 0_R \) and universal
element \( 1_R \). The elements of \( B_{\lor R} \) are ordered by inclusion, denoted by \( \leq \).
(b) If the products $P \circ R$ and $Q \circ R$ are defined, so is $P \circ Q^\sim$. If the products $P \circ Q$ and $P \circ R$ are defined, so is $Q^\sim \circ R$. If $Q \circ R$ exists, so does $Q \circ P$ for every $P \in B_{p,R}$.

(c) Composition is associative: $P \circ (Q \circ R) = (P \circ Q) \circ R$.

(d) There are elements $\text{id}_R$ and $i_R$ associated to every relation $R \in B$. $\text{id}_R$ behaves as a right identity and $i_R$ as a left identity for $B_{p,R}$.

(e) The Schröder rule $P \circ Q \leq R \iff P^\sim \circ R \leq -Q \iff -R \circ Q^\sim \leq -P$ holds whenever one of the three expressions is defined.

(f) $1 \circ R \circ 1 = 1$ iff $R \neq 0$ (Tarski rule).

If $R^\sim \in B_{p,R}$, then $R$ is said to be homogeneous. If all $R \in A$ have the same type, the operations are all total and $A$ itself is said to be homogeneous.

For simplicity, the universal, zero, and identity elements are all denoted by 1, 0, $\text{id}$, respectively. One can use subscripts to make the typing explicit, but this will not be necessary here. The precedence of the relational operators, from highest to lowest, is the following: $\sim$ and $\leq$ bind equally, followed by $\circ$, followed by $\land$ and finally by $\lor$. The scope of $\lor_i$ and $\land_i$ goes to the right as far as possible. $R^\sim$ is called the converse of $R$. Another operation that occurs in this article is the reflexive transitive closure $R^\ast$. It satisfies the well-known laws

$$R^\ast = \bigvee_{i \geq 0} R^i \quad \text{and} \quad R^\ast = \text{id} \lor R \lor R^\ast = \text{id} \lor R^\ast \circ R,$$

where $R^0 = \text{id}$ and $R^{i+1} = R \circ R^i$. From Definition 2.1, the usual rules of the calculus of relations can be derived (see, e.g., [9, 14, 42]). We assume these rules to be known and simply recall a few of them. The notion of Galois connections is very important in what follows, there are many definitions of Galois connections [1]. We choose the following one [6].

**Definition 2.2.** Let $(S, \leq_S)$ and $(S', \leq_{S'})$ be two preordered sets. A pair $(f, g)$ of functions, where $f : S \rightarrow S'$ and $g : S' \rightarrow S$, forms a Galois connections iff the following formula holds for all $x \in S$ and $y \in S'$.

$$f(x) \leq_{S'} y \iff x \leq_S g(y).$$

The function $f$ is called the lower adjoint and $g$ the upper adjoint.

3. Monotypes and related operators.

3.1. **Monotypes.** In the calculus of relations, there are two ways for viewing sets as relations; each of them has its own advantages. The first is via vectors: a relation $x$ is a vector [42] iff $x = x \circ 1$. The second way is via monotypes [6]: a relation $a$ is a monotype iff $a \leq \text{id}$. The set of monotypes $\{a \mid a \in B_{p,R}\}$, for a given $R$, is a complete Boolean lattice. We denote by $a^\sim$ the monotype complement of $a$.

3.2. **Domain and codomain operators.** The domain and codomain of a relation $R$ can be characterized by the vectors $R \circ 1$ and $R^\sim \circ 1$, respectively [21, 42]. They can also be characterized by the corresponding monotypes. In this paper, we take the last approach. In what follows we formally define these operators and give some of their properties.

**Definition 3.1.** The domain and codomain operators of a relation $R$, denoted respectively by $R^\leq$ and $R^\geq$, are the monotypes defined by the equations

(a) $R^\leq = \text{id} \land R \circ 1,$
(b) $R^\succ = \text{id} \land 1 \circ R$. 

These operators can also be characterized by Galois connections (see [6]). For each relation $R$ and each monotype $a$,

\[
R^\prec \leq a \iff R \leq a \circ 1,
\]

\[
R^\succ \leq a \iff R \leq 1 \circ a.
\]

The domain and codomain operators are linked by $R^\succ = R^\prec \succ$.

### 3.3. Monotype residuals.

**Definition 3.2.** Let $R$ be a relation and $a$ be a monotype. The **monotype right residual** and monotype left residual of $a$ by $R$ (called factors in [6]) are defined respectively by

\[ a \setminus \cdot R := ((1 \circ a)/R)^\succ, \]

\[ R \setminus \cdot a := (R \setminus (a \sqcap 1))^\prec. \]

An alternative characterization of residuals can also be given by means of a Galois connection as follows [5]:

\[ b \leq a \setminus \cdot R \iff (b \sqcup R)^\succ \leq a, \]

and

\[ b \leq R \setminus \cdot a \iff (R \circ b)^\prec \leq a. \]

Since we do not use the operator $\setminus \cdot$ in the sequel, we need the next property of $\setminus \cdot$.

\[ a \leq b \Rightarrow a \setminus \cdot R \leq b \setminus \cdot R, \quad (1) \]

We now give an additional property of the monotype complement and monotype residual operators.

\[ a^\sim = 0 \setminus \cdot a. \quad (2) \]

In the following, we have to use exhaustively the complement of the domain of a relation $R$, i.e., the monotype $a$ such that $a = R^\prec \sim$. To avoid the notation $R^\prec \sim$, we adopt the Notation

\[ R^\prec := R^{\prec \sim}. \quad (3) \]

Because we assume our relation algebra to be complete (Definition 2.1), least and greatest fixed points of monotonic functions exist. We cite [15] as a general reference on fixed points.

Let $f$ be a monotonic function. The following properties of fixed points are used below:

\[ \mu f = \bigwedge \{X \mid f(X) = X\} = \bigwedge \{X \mid f(X) \leq X\}, \quad (4) \]

\[ \nu f = \bigvee \{X \mid f(X) = X\} = \bigvee \{X \mid X \leq f(X)\}, \]

\[ \mu f \leq \nu f, \]

\[ f(Y) \leq Y \Rightarrow \mu f \leq Y, \]

\[ Y \leq f(Y) \Rightarrow Y \leq \nu f. \]

In this subsection, we describe notions that are useful for the description of the set of initial states of a program for which termination is guaranteed. These notions are **progressive finiteness** and the **initial part** of a relation.

### 3.4. Progressive finiteness of a relation.

A relation $R$ is progressively finite in terms of points iff there are no infinite chains $s_0, \ldots, s_i$ such that $s_i R s_{i+1} \forall i, i \geq 0$. Let there be no points set $y$ which are the starting points of some path of infinite length. The least set of points which are the starting points of paths of finite length i.e. from which we can proceed only finitely many steps is called **initial part** of $R$ denoted by $I(R)$. This topic is of interest in many areas of computer science, mathematics and is related to recursion and induction principle.
Definition 3.3. (a) The initial part of a relation $R$, denoted $I(R)$, is given by $I(R) = \bigwedge \{ a \mid a \leq \text{id} : a \# R = a \} = \bigwedge \{ a \mid a \leq \text{id} : a \# R \leq a \} = \mu(a : a \leq \text{id} : a \# R)$, where $a$ is a monotype.

(b) A relation $R$ is said to be progressively finite [42] if $I(R) = \text{id}$.

The description of $I(R)$ by the formulation $a \# R = a$ shows that $I(R)$ exists, since $(a \mid a \leq \text{id} : a \# R)$ is monotonic in the first argument (by Equation (1)) and because the set of monotypes is a complete lattice, it follows from the fixed point theorem of Knaster and Tarski that this function has a least fixed point. Progressive finiteness of a relation $R$ is the same as well-foundedness of $R^\omega$.

$I(R)$ is a monotype. In a concrete setting, $I(R)$ is the set of monotypes which are not the origins of infinite paths (by $R$). Using Equations 4 and Boolean laws, one has: A relation $R$ is progressively finite iff for a monotype $a, a \leq (R \circ a)^\prec \Rightarrow a = 0$ equivalently $\nu(a : a \leq \text{id} : (R \circ a)^\prec) = 0$ equivalently $\mu(a : a \leq \text{id} : a \# R) = \text{id}$. In [5], it is shown that the following is also equivalent:

The next theorem involves the function $w_a(X) := Q \vee P \circ X$, which is closely related to the description of iterations. The theorem highlights the importance of progressive finiteness in the simplification of fixed point-related properties.

Theorem 3.4. Let $f(X) := Q \vee P \circ X$ be a function. If $P$ is progressively finite, the function $f$ has a unique fixed point which means that $\nu(f) = \mu(f) = P^* \circ Q$ [5]:

4. A demonic refinement ordering. We now define the refinement ordering (refinement ordering) we will be using in the sequel. This ordering induces a complete join semilattice, called a demonic semilattice. The associated operations are demonic join ($\sqcup$), demonic meet ($\sqcap$) and demonic composition ($\circ$). We give the definitions and needed properties of these operations, and illustrate them with simple examples. For more details on relational demonic semantics and demonic operators, see [6, 7, 8, 9, 20].

We will give the rationale behind the definition of refinement called the refinement ordering.

If we consider a relation $R$ as a specification of the input-output behavior of a program $p$, then $p$ will be totally correct with respect to $R$ if:

- for any input $i$ in the domain of $R$, $i'$ is a possible output of $p$ only if $(i, i') \in R$,
- $p$ always terminates for any input belonging to the domain of $R$ [37]. For an input that does not belong to the domain of specification $R$, program $p$ may return any result or return no result; that is, the specifier does not care what happens following the submission of such an input.

Definition 4.1. We say that a relation $Q$ refines a relation $R$ [38], denoted by $Q \subseteq R$, iff

$$R^\prec \circ Q \leq R \quad \text{and} \quad R^\prec \leq Q^\prec.$$  

In other words, $Q$ refines $R$ if and only if
• $R^< \leq Q^<$: this means that $Q$ must assign a result to any state to which $R$ also assigns a result;
• $R^< \circ Q \leq R$: the prerestrición of $Q$ to the domain of $R$ is included in $R$:
  this means that $Q$ must not produce results not allowed by $R$ for those states
  that are in the domain of $R$.

Thus, for instance,
\[
\begin{pmatrix}
1 & 0 & 0 \\
1 & 1 & 0
\end{pmatrix} \subseteq \begin{pmatrix} 1 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}, \quad \text{but } \begin{pmatrix} 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0
\end{pmatrix} \not\subseteq \begin{pmatrix} 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

(These Boolean matrices represent relations over sets by the well-known correspondence).

**Proposition 1.** Let $Q$ and $R$ be relations, then

(a) The greatest lower (wrt $\sqsubseteq$) of $Q$ and $R$ is,

\[ Q \sqcup R = Q^< \circ R^< \circ (Q \lor R), \]

If $Q^< = R^<$ then we have $\sqcup$ & $\lor$ coincide i.e $Q \sqcup R = Q \lor R$.

For example, with
\[ Q := \{(0,0),(1,0)\} \quad \text{and} \quad R := \{(1,1),(2,0)\}, \]
we have

\[ Q \sqcup R = \{(1,0),(1,1)\}. \]

This operation corresponds to a demonic non-deterministic choice, since

the possibility of failure (like in state 0, as there is one relation (namely, $R$) that
produces no result for this state) is reflected in the result. On the other hand, for state 1,
both relations produce a result and both results are results of $Q \sqcup R$ for state 1.

(b) If $Q$ and $R$ satisfy the condition $Q^< \land R^< = (Q \land R)^<$, their least upper bound is

\[ Q \sqcap R = Q \land R \lor Q^< \circ R \lor R^< \circ Q, \]

otherwise, the least upper bound does not exist. If $Q^< \land R^< = 0$ then we have $\sqcap$ & $\land$ coincide i.e $Q \sqcap R = Q \land R$.

For the proofs see [12, 20]. And here is an example of these operations:

\[
\begin{pmatrix}
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix} \sqcup \begin{pmatrix} 0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1
\end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\
1 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}.
\]

This operation corresponds to a demonic non-deterministic choice, since

the possibility of failure (row 3 of the first matrix or row 1 of the second) is reflected in
the result. For the middle row, failure is not possible, and the set of allowed results is
the union of the results of the two operands.

**Definition 4.2.** The demonic composition of relations $Q$ and $R$ [6] is

\[ Q \triangleright R = (R^< \bullet Q) \circ Q \circ R. \]

For concrete relations, this definition is read as follows: a pair $(i,i')$ belongs to $Q \triangleright R$ if and only if it belongs to $Q \circ R$ and there is no possibility of reaching, from
$i$, by $Q$, an element $i''$ that does not belong to the domain of $R$. For example, with
\[ Q := \{(0,0),(0,1),(1,2)\} \quad \text{and} \quad R := \{(0,0),(2,3)\}, \]
Proposition 4. Let $R$, $S$ be relations and $A$ be a monotype we have:

\begin{align*}
(a) \quad R^< \circ 1 &= R \circ 1, \\
(b) \quad (R \circ 1)^< &= R^<, \\
(c) \quad R^< \circ R &= R, \\
(d) \quad (R \circ S)^< &= (R \circ S)^<, \\
(e) \quad a^< &= a, \\
(f) \quad (a \circ R)^< &= a \circ R^<, \\
(g) \quad (R \lor S)^< &= R^< \lor S^<.
\end{align*}

5. Relational diagrams. In the following, we will give the formal definition of a diagram and different types of diagrams. We define a diagram as being a quadruple constructed of a relation, a set of monotypes mutually disjoint; they have a role identical to the vertices of a graph and also of two particular monotypes characterizing the input and the output of the diagram. The word diagram is well justified by the use of matrices and graphs.

Even though we don’t use matrices and graphs in our formal treatment (already used in our anterior work \[46, 47\]), we use them in our informal and intuitive explications.

How the matrices can be obtained from diagrams is intuitively obvious, and we refer to \[41, 42, 46, 47\] for a rigorous treatment.

The set of matrices whose entries are relations constitutes a relation algebra \[41, 42\], with the relational operations defined as follows. Let $R$, $S$ be matrices whose entries belong to the same homogeneous algebra. Then,

\begin{align*}
(R \lor S)[i, j] &= R[i, j] \lor S[i, j], \\
(R \land S)[i, j] &= R[i, j] \land S[i, j], \\
\overline{R}[i, j] &= \overline{R[i, j]}, \quad R[i, j]^\lor = (R[j, i])^\lor, \\
(R \circ S)[i, j] &= \lor_k R[i, k] \circ S[k, j],
\end{align*}

where $R[i, j]$ denotes entry $i, j$ of matrix $R$. Of course, $R \lor S$ and $R \land S$ exist only if matrices $R$ and $S$ have the same dimension; the composition $R \circ S$ exists only if the number of columns of $R$ is the same as the number of rows of $S$. The entries of the identity matrix (which is square) are $0$, except those of the diagonal, which are $1$. The entries of the zero matrix are $\emptyset$ and those of the universal matrix are $1$.

Definition 5.1. Let $\mathcal{A}$ be a homogeneous relational algebra.

(a) A quadruple $\mathcal{P} = (P, C, e, s)$ is a diagram on $\mathcal{A}$ if:

- $P$ is a relation of $\mathcal{A}$, called the associated relation of diagram $\mathcal{P}$,
- $C$ is a set of monotypes instead to be vectors as it has already been done in \[47\]. The monotypes are disjoint from each other, verifying the condition : $(\lor C) \circ P \circ (\lor C) = P$,
- $e$ (entry) and $s$ (sortie), which means exit in french, are monotypes $(e, s \in C)$ called respectively the input relation and the output relation of the diagram $\mathcal{P}$.

(b) A diagram $\mathcal{P}_1 = (P_1, C, e_1, s_1)$ is a sub-diagram of diagram $\mathcal{P} = (P, C, e, s)$ iff:

\[ P_1 \leq P \quad \text{and} \quad e_1^\lor \circ s_1^\lor \circ (\lor C) \circ P = P \circ e_1^\lor \circ s_1^\lor \circ (\lor C) = 0. \]

In other words, a sub-diagram is a diagram such that its associated relation is included in the associated relation of the principal diagram. When the principal diagram is executed, the sub-diagram is also executed. The control begins by the input node $e_1$ to $\mathcal{P}_1$ and terminates by the output node $s_1$. The diagram $\mathcal{P}_1$ communicates no information to the diagram $\mathcal{P}$ through the internal nodes of the diagram $\mathcal{P}_1$.

Let us give an example that illustrates definition 5.1(a). Consider the loop :
\textbf{do n > 0 → n := n - 10 od},

where \(n\) takes its values on the set of natural numbers \(\mathbb{N}\), which is the set of states. By following the same approach as Schmidt and Ströhlein [42], this program can be represented by the following graph:

![Graph](attachment:image.png)

The node 0 is the input node and the node 2 is the output node. Each arc is labeled with a relation on \(\{0, 1, 2\} \times \mathbb{N}\) calculated by the program between the nodes related by this arc. These relations are:

- \(P_1 := \{((0, n), (1, n')) | n > 0 \land n' = n\}\),
- \(P_2 := \{((1, n), (0, n')) | n \in \mathbb{N} \land n' = n - 10\}\),
- \(P_3 := \{((0, n), (2, n')) | n \leq 0 \land n' = n\}\).

For example, the relation on \(\{0, 1, 2\} \times \mathbb{N}\) calculated by the loop body between the nodes 1 and 0 is \(P_2\). The representation matrix of this graph is:

\[
\begin{pmatrix}
0 & 1 & 2 \\
0 & P_1 & P_3 \\
2 & P_2 & \emptyset & \emptyset & \emptyset
\end{pmatrix}
\]

We will give the relations characterizing the diagram \(\mathcal{P} = (P, C, e, s)\).

- The associated relation of this diagram is \(P = P_1 \lor P_2 \lor P_3\). Then, \(P\) is a relation of type \(\{0, 1, 2\} \times \mathbb{N} \leftrightarrow \{0, 1, 2\} \times \mathbb{N}\).
- \(C = \{e, a, s\}\) where \(e = \{((0, n), (0, n')) | n' = n\}\), \(a = \{((1, n), (1, n')) | n' = n\}\) and \(s = \{((2, n), (2, n')) | n' = n\}\).

We can easily verify that \(e, a, s\) are of type \(\{0, 1, 2\} \times \mathbb{N} \leftrightarrow \{0, 1, 2\} \times \mathbb{N}\), are partial identities disjoint from each other and verify \((e \lor a \lor s) \sqcap P \circ (e \lor a \lor s) = P\).

We distinguish two types of diagrams: elementary and compound diagrams.

**Definition 5.2.**

(a) A diagram \(\mathcal{P} = (P, C, e, s)\) is \textit{atomic} iff \(C = \{e, s\}\) and \(P = e \sqcap P \circ s\).

The graph and the matrix associated with this diagram are:

\[
\begin{pmatrix}
e & s \\
\emptyset & P & \emptyset & \emptyset & \emptyset
\end{pmatrix}
\]

An atomic diagram consists of a unique atomic step \(i.e.\) the transition between the input node and the output node is formed in exactly one step (there is only one input which is different from \(\emptyset\) in the matrix).

(b) A diagram \(\mathcal{P} = (R, C, e, s)\) is a \textit{sequence diagram} iff:

\(P = P_1 \lor P_2,\ C = \{e, a, s\},\ P = e \sqcap P_1 \circ a\ and\ P_2 = a \sqcap P_2 \circ s.\)
This diagram can be represented by the following graph and matrix:

\[
P = a \begin{pmatrix} \emptyset & P_1 & \emptyset \\ \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & P_2 \end{pmatrix}
\]

(5)

The graph and also the matrix illustrate well the use of the word “sequence.” We note that a sequence diagram is the sequence of two atomic diagrams such that the output relation of the first one is equal to input relation of the second one.

(c) A diagram \( P = (R, C, e, s) \) is a branching diagram iff

\[
R = G_1 \lor P \lor G_2 \lor Q, \quad C = \{e, a, b, s\}, \quad G_1 = e \Box G_1 \circ a, \quad P = a \Box P \circ s, \quad G_2 = e \Box G_2 \circ b \quad \text{and} \quad Q = b \Box Q \circ s.
\]

The graph and the associated matrix representing this diagram are:

\[
R = a \begin{pmatrix} e & a & b & s \\ \emptyset & G_1 & G_2 & \emptyset \\ \emptyset & \emptyset & \emptyset & P \\ \emptyset & \emptyset & \emptyset & Q \\ s & \emptyset & \emptyset & \emptyset \end{pmatrix}
\]

(6)

In a branching diagram, the execution begins in the input node and the output node is formed in a nondeterministic way in two different branches and each one is composed of two atomic steps. We notice that, if \( G_2 = Q = b = \emptyset \) (or the symmetric case), then \( P \) is transformed into a sequence diagram.

(d) A diagram \( W = (W, C, e, s) \) is a loop diagram iff

\[
W = P \lor Q, \quad P = e \Box P \circ s, \quad Q = e \Box Q \circ s
\]

The graph and the matrix representing the loop diagram \( P \) are:

\[
P = e \begin{pmatrix} P & Q \\ \emptyset & \emptyset \end{pmatrix}
\]

(7)

In a loop diagram, \( P \) is applied until \( Q \) can be applied.

(e) A diagram is elementary if it is an atomic, a sequence, branching or a loop diagram.

(f) A diagram is compound if it is not elementary.

6. Demonic input-output relation of compound diagrams. During the execution of a program in an input state, by considering a demonic point of view (if there is a possibility for the program not to terminate normally then it will not terminate normally), three cases may happen: normal termination, abnormal
termination and infinite loops. As our goal is to define formally the input-output relation of a diagram by supposing its worst execution, we have to consider these three cases together at the same time. For more details see [47]. In the following, the expression “input-output relation” means the demonic input-output relation. The input-output relation of a diagram $\mathcal{P}$ is given by a relation $\mathcal{E}(\mathcal{P})$ where $\mathcal{E}$ is a function from the set of diagrams to a relational algebra, which associates to each diagram $\mathcal{P}$ the relation $\mathcal{E}(\mathcal{P})$ given by:

$$\mathcal{E}(\mathcal{P}) = e \circ (T \land \mathcal{I}(\mathcal{P})) \circ s,$$

where $T := P^* \circ P^*$. (8)

In the following, we will consider the case of compound diagrams. Suppose that we have a certain compound diagram $\mathcal{P} = (\bigvee_{i=1}^{n} \mathcal{P}_i, C, e, s)$ composed of elementary sub-diagrams (Definition 5.1) $\mathcal{P}_i = (\mathcal{P}_i, C_i, c_i, s_i)$. Instead of calculating the input-output relation $\mathcal{E}(\mathcal{P})$ of diagram $\mathcal{P}$ directly by applying Equation 8, which is a laborious task, we will prove that the relation $\mathcal{E}(\mathcal{P})$ is equal to the input-output relation of the diagram obtained from diagram $\mathcal{P}$ by replacing each sub-diagram $\mathcal{P}_i$ by its input-output relation $\mathcal{E}(\mathcal{P}_i)$. The process continues until we obtain elementary diagrams for which we apply the results given in [47]. Let us give an example: Let $\mathcal{P} = (P, C, e, s)$ be a diagram constructed of two sub-diagrams $\mathcal{P}_1 = (P_1, C, e, c)$ and $\mathcal{P}_2 = (P_2, C, e, s)$, as illustrated by Figure 1. The sub-diagrams $\mathcal{P}_1$ and $\mathcal{P}_2$ may even be compound diagrams. Instead of calculating directly the input-output relation $\mathcal{E}(\mathcal{P})$ of diagram $\mathcal{P}$ by applying Equation 8, we want to show that $\mathcal{E}(\mathcal{P}) = \mathcal{E}(\mathcal{P}')$, with

$$\mathcal{P}' = (\mathcal{E}(\mathcal{P}_1) \lor \mathcal{E}(\mathcal{P}_2), C, e, s), \quad C = \{e, c, s\}, \quad \mathcal{E}(\mathcal{P}_1) = e \circ \mathcal{E}(\mathcal{P}_1) \circ c, \quad \mathcal{E}(\mathcal{P}_2) = e \circ \mathcal{E}(\mathcal{P}_2) \circ s.$$

We remark that $\mathcal{P}'$ is a sequence diagram (see the graph of Figure 1). By applying Equation the results in previous work [47, 48] to diagram $\mathcal{P}'$, we obtain $\mathcal{E}(\mathcal{P}') = \mathcal{E}(\mathcal{P}_1) \lor \mathcal{E}(\mathcal{P}_2)$, whence we deduce $\mathcal{E}(\mathcal{P}) = \mathcal{E}(\mathcal{P}_1) \lor \mathcal{E}(\mathcal{P}_2)$. It is clear that behind this result a whole preliminary formal treatment is hidden. We also have to consider the other types of diagrams, i.e. the branching diagram and the loop diagram. Instead of treating each case separately, we will give a general result which regroups all the cases. We will restrict ourselves to the following constraints that we explain below:

**Hypotheses 1.** Let $\mathcal{R} = (R, C, e, s)$, be a diagram in which we identify a sub-diagram $\mathcal{P} = (P, C_P, a, c)$ (Figure 2) such that:

| (a) | $R = P \lor Q$, |
| (b) | $C = \{a, b, c, d\}$, |
| (c) | $P = (a \lor b) \circ P \circ a \lor b \lor c$, |
| (d) | $Q = (c \lor d) \circ Q \circ a \lor c \lor d$, |
| (e) | $C_P = \{a, b, c\}$, |
| (f) | $e \leq a \lor c \lor d$, |
| (g) | $s \leq c \lor d$. |

Our goal is to decompose the relation $R$ to extract a sub-diagram $\mathcal{P}$ of diagram $\mathcal{R}$. The input states of $\mathcal{P}$ are represented by $a$, the output states by $c$ and the states which are neither input nor output states, which we call internal states, are represented by $b$. These three elements are mutually disjoint, in other words
a ∧ b = c ∧ a = c ∧ b = 0, also c □ P = 0 (i.e. c □ P = 0) which means that c is terminal. The monotype d represents all the states of R except those of a, b, c. Therefore, d □ P = 0, P □ P = 0 and d ∧ a = d ∧ b = d ∧ c = 0. By considering these conditions and the fact that P is the associated relation of the sub-diagram P, we have P = (a ∧ b) □ P ∪ (a ∧ b ∧ c). We suppose that the internal states of P cannot be used to transmit the control from P to Q or from Q to P, which gives b □ Q = Q □ b = 0. We suppose also that the states of a are used as entries to P only, then a □ Q = 0. We obtain :

Q = (c ∨ d) □ Q ∪ (a ∨ c ∨ d).

Remark 1. Each compound diagram R = (\biglor_{i=1}^{n} P, C, e, s) composed of sub-diagrams P = (P, C, e_i, s_i) can be brought back to Hypotheses 1 by identifying a sub-diagram P_k = (P_k, C_k, e_k, s_k), where 1 ≤ k ≤ n, and by taking

(a) P := P_k,
(b) Q := \biglor_{i=1}^{n} P_i, i \neq k,
(c) a := e_k,
(d) b := (e_k ∨ s_k) □ (\biglor C_k),
(e) c := s_k,
(f) d := (\biglor C_k) □ (\biglor_{i=1}^{n} C_i),
(g) e := e.

In the following, we will present the main result of this paper. It is sufficient to give an explication of this theorem and also its application in Section 7.

Theorem 6.1. Let R := (R, C, e, s) and P := (P, C, a, c) verify Hypotheses 1. We have E(R) = E(R'), where R' := (E(P) ∪ Q, C, e, s).

This means that the input-output relation of diagram R = (P ∪ Q, C, e, s) is equal to the input-output relation of diagram (E(P) ∪ Q, C, e, s) ; the associated relation of diagram P has been replaced by the input-output relation of the last one.
(see the matrix below). We note that the diagram \((\mathcal{E}(P), \{a, c\}, a, c)\) is an atomic
diagram (Definition 5.1(a)).

\[
R' = \begin{pmatrix}
a & b & c & d \\
a & 0 & 0 & \mathcal{E}(P) & 0 \\
b & 0 & 0 & 0 & 0 \\
c & Q_1 & Q_2 & Q_3 & \\
d & Q_4 & Q_5 & Q_6 & \\
\end{pmatrix}
\]

The same reasoning may be applied to the rest of diagram \(\mathcal{R} \); it is enough to
identify in the diagram \(\mathcal{R}'\) a sub-diagram which verifies Hypotheses 1 and we apply
this until we obtain elementary diagrams for which we apply the results of the
second chapter of the Thesis [47, 48] and also in [13].

In the following section, we will show how to apply Theorem 6.1 to certain types
of diagrams.

7. Applications. In this section, we will illustrate the application of Theorem 6.1
calculating the input-output relation of compound diagrams which may be reduced
to sequence diagrams.

Let the diagrams \(P, P_1\) and \(P_2\) be such that

\[
P = (P_1 \lor P_2, C, e_1, s_2), \quad P_1 = (P_1, C_1, e_1, s_1) \quad \text{and} \quad P_2 = (P_2, C_2, s_1, s_2),
\]

where \(C = C_1 \lor C_2, \quad (\forall C_1) \land (\forall C_2) = s_1 \quad \text{and} \quad s_1 \sqcap P_1 = 0.\)

Suppose that diagrams \(P_1\) and \(P_2\) are not atomic. Our aim is to calculate the
input-output relation \(\mathcal{E}(P)\) of diagram \(P\) by applying Theorem 6.1 and the results
from [47]. To do that, we reduce \(P_1\) and \(P_2\) to obtain a sequence diagram (Definition 5.1).

Substitution 1. Consider the following substitutions where the variables on the
left side are those of Theorem 6.1 :

\[
\begin{align*}
(a) & \quad a := e_1, \\
(b) & \quad b := e_1 \lor s_1 \sqcap (\forall C_1), \\
(c) & \quad c := s_1, \\
(d) & \quad d := s_1 \sqcap (\forall C_2), \\
(e) & \quad P := P_1, \\
(f) & \quad Q := P_2, \\
(g) & \quad e := e_1, \\
(h) & \quad s := s_2.
\end{align*}
\]

The relation \((e_1 \lor s_1) \sqcap (\forall C_1)\) represents the states of the diagram \(P_1\) which are
different from input and output states (the internal states). By using \(c \sqcap P = 0\)
and also the properties verified by \(a, b\) (Hypothesis 1 and Substitution 1), we obtain
\((a \lor b) \sqcap P \lor (a \lor b \lor c) = (\forall C_1) \sqcap P \lor (\forall C_1) = P.\)

It is not difficult to verify that Hypotheses 1 are satisfied. By Theorem 6.1, we
have then

\[
\mathcal{E}(P) = \mathcal{E}(\mathcal{E}(P_1) \lor P_2, C', e, s), \quad \text{where} \quad C' = C_2 \lor \{e_1\}.
\]

By hypotheses, \(P_2 = (P_2, C_2, s_1, s_2)\) is a diagram; we will proceed in the same way
and apply once more Theorem 6.1 to the new diagram \((\mathcal{E}(P_1) \lor P_2, C, e, s)\) to obtain
\(\mathcal{E}(P) = \mathcal{E}(P'),\) where \(P' = (\mathcal{E}(P_1) \lor \mathcal{E}(P_2), C', e_1, s_2),\) where \(C' = \{e_1, s_1, s_2\}.\) By
the definitions of \(E, \mathcal{P}_1\) and \(\mathcal{P}_2,\) we have \(\mathcal{E}(P_1) = e_1 \sqcap \mathcal{E}(P_1) \circ s_1\) and \(\mathcal{E}(P_2) = s_1 \sqcap \mathcal{E}(P_2) \circ s_2.\) So, \(P'\) is a sequence diagram (Definition 5.1). By applying previous
results, we obtain :

\[
\mathcal{E}(P) = \mathcal{E}(P_1) \sqcap \mathcal{E}(P_2).
\]
8. Conclusion. We have presented certain notions concerning relation algebra and relational diagrams in Sections 2 and 5 and also the refinement order in Section 4. We have shown how to give a generic demonic input-output relation definition (Equation 8), based on the concept of relational diagrams [42, 47]. This definition has been used to calculate the demonic input-output relation of an arbitrary instruction, which we have done in the case of the sequence (Equation 5). Our main result is Theorem 6.1, where we showed that the input-output relation of compound diagram is equal to that of the diagram in which each sub-diagram has been replaced by its input-output relation. This process is repeated until we obtain elementary diagrams to which we apply the results given in [47]. To prove Theorem 6.1, we needed many intermediate results. Some of them have been demonstrated by the generalisation of Mills’ theorem [18, 46, 47, 48, 49, 50, 51, 52]. In Section 7, we have applied Theorem 6.1 to a compound diagram corresponding to a particular case (sequence) of language of guarded commands.

The approach to demonic input-output relation presented here is not the only possible one. In [10, 31, 32, 33], the infinite looping has been treated by adding to the state space a fictitious state $\bot$ to denote nontermination. In [9, 25, 30, 35, 36, 39, 54, 55, 57], the demonic input-output relation is given as a pair (relation, set). The relation describes the input-output behavior of the program, whereas the set component represents the domain of guaranteed termination.

We note that the preponderant formalism employed until now for the description of demonic input-output relation is the wp-calculus. For more details see [2, 3, 23, 56, 58].

Acknowledgments. This research project was supported by a grant from the Research Center of the Center for Female Scientific and Medical Colleges, Deanship of Scientific Research, King Saud University.

REFERENCES

[1] C. Aarts, R. Backhouse, P. Hoogendijk, E. Voermans and J. van der Woude, A Relational Theory of Datatypes, Department of Computing Science, Eindhoven University of Technology, 1992.
[2] R. J. R. Back, On the correctness of refinement in program development, Thesis, Department of Computer Science, University of Helsinki, 1978.
[3] R. J. R. Back, Combining angels, demons and miracles in program specifications, Theoretical Computer Science, 100 (1992), 365–383.
[4] R. J. R. Back, A continuous semantics for unbounded nondeterminism, Theoretical Computer Science, 23 (1983), 187–210.
[5] R. C. Backhouse and H. Doornbos, Mathematical Induction Made Calculational, Computing Science Note 94/16, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, 1994.
[6] R. C. Backhouse and J. van der Woude, Demonic operators and monotype factors, Mathematical Structures in Computer Science, 3 (1993), 417–433. Also: Computing Science Note 92/11, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, 1992.
[7] R. Berghammer, Relational Specification of Data Types and Programs, Technical report 9109, Fakultät für Informatik, Universität der Bundeswehr München, Germany, September 1991.
[8] R. Berghammer and G. Schmidt, Relational specifications, in Algebraic Logic, Banach Center Publications (ed. C. Rauszer), 28, Polish Academy of Sciences, 1993, 167–190.
[9] R. Berghammer and H. Zierer, Relational algebraic semantics of deterministic and nondeterministic programs, Theoretical Computer Science, 43 (1986), 123–147.
[10] J. Bharat and T. Pallavi, Compositional semantics for diagrams using constrained objects, in International Conference No 2, Callaway Gardens GA, ETATS-UNIS (18/04/2002), Lecture Notes in Computer Science, Vol. 2317, 2002, 94–96.

[11] C. Böhm, On a family of Turing machines and the related programming languages, ICC Bull., 3 (1964), 185–194.

[12] N. Boudriga, F. Elloumi and A. Mili, On the lattice of specifications: Applications to a specification methodology, Formal Aspects of Computing, 4 (1992), 544–571.

[13] C. Brink, W. Kahl and G. Schmidt, eds., Relational Methods in Computer Science, Springer, 1997.

[14] L. H. Chin and A. Tarski, Distributive and modular laws in the arithmetic of relation algebras, University of California Publications, 1 (1951), 341–384.

[15] B. A. Davey and H. A. Priestley, Introduction to Lattices and Order, Cambridge Mathematical Textbooks, Cambridge University Press, Cambridge, 1990.

[16] W. P. De Roever, Recursive Program Schemes: Semantics and Proof Theory, Math. Centrum Tracts, Amsterdam, Theses, 1974.

[17] J. Desharnais, B. Möller and F. Tchier, Kleene under a modal demonic star, Journal of Logic and Algebraic Programming, 66 (2006), 127–160.

[18] J. Desharnais, B. Möller and F. Tchier, Kleene under a demonic star, in 8th International Conference on Algebraic Methodology And Software Technology (AMAST 2000), May 2000, Iowa City, Iowa, USA, Lecture Notes in Computer Science, 1816, Springer-Verlag, 2000, 355–370.

[19] J. Desharnais, N. Belkhiter, S. B. M. Sghaier, F. Tchier, A. Jaoua, A. Mili and N. Zaguia, Embedding a demonic semilattice in a relation algebra, Theoretical Computer Science, 149 (1995), 333–360.

[20] J. Desharnais, F. Tchier and R. Khédri, Demonic Relational Semantics of Sequential Programs, Rapport de recherche DIUL-RR-9406, Département d’informatique, Université Laval, Québec, QC, G1K 7P4, Canada, 1994.

[21] J. Desharnais, A. Jaoua, F. Mili, N. Boudriga and A. Mili, A Relational division operator: The conjugate kernel, Theoretical Comput. Sci., 114 (1993), 247–272.

[22] J. Desharnais, Abstract Relational Semantics, Ph D. thesis, School of Computer Science, Univ. McGill, Montréal, 1989.

[23] E. W. Dijkstra, A Discipline of Programming, Prentice-Hall, 1976.

[24] H. Doornbos, Reductivity, Science of Computer Programming, 26 (1996), 217–236.

[25] H. Doornbos, A relational model of programs without the restriction to Egli-Milner monotone constructs, in IFIP Transactions, A-56: North-Holland, 1994, 363–382.

[26] H. Doornbos, R. Backhouse and J. van der Woude, A calculational approach to mathematical induction, Theoretical Computer Science, 179 (1997), 103–135.

[27] R. W. Floyd, Assigning meanings to programs, Proceedings AMS Symposium in Applied Mathematics, 19 (1967), 19–32.

[28] M. Frappier, A Relational Basis for Program Construction by Parts, Dept. of Computer Science, University of Ottawa, 1994.

[29] C. Gunter, Semantics of Programming Languages, MIT Press, 1992.

[30] H. Riis Nielsen and F. Nielsen, Semantics with Applications: An Appetizer, Undergraduate Topics in Computer Science, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[31] C. A. R. Hoare and J. He, The weakest prespecification, Part I, Fundamenta Informaticae IX, 9 (1986), 51–84.

[32] C. A. R. Hoare and J. He, The weakest prespecification, Part II, Fundamenta Informaticae IX, 9 (1986), 217–251.

[33] C. A. R. Hoare, et al., Laws of programming, Communications of the ACM, 30 (1987), 672–686.

[34] Y. I. Ianov, On the equivalence and transformation of program schemes, Dokl. Akad. Nauk, 1 (1958), 8–12.

[35] J. S. Reich and J. L. Jacob, Relational Denotational Semantics of the While Language, Department of Computer Science, University of York, UK.

[36] R. D. Maddux, Relation-algebraic semantics, Theoretical Computer Science, 160 (1996), 1–85.

[37] A. Mili, A relational approach to the design of deterministic programs, Acta Informatica, 20 (1983), 315–328.
A. Mili, J. Desharnais and F. Mili, Relational heuristics for the design of deterministic programs, *Acta Informatica*, **24** (1987), 239–276.

D. L. Parnas, A generalized control structure and its formal definition, *Communications of the ACM*, **26** (1983), 572–581.

J. Riguet, Programmation et théorie des catégories, in *Proc. ICC Symp. Symbolic Languages in Data Processing*, Gordon and Breach, New York, 1962, 83–98.

G. Schmidt, Programs as partial graphs I: Flow equivalence and correctness, *Theoretical Computer Science*, **15** (1981), 1–25.

G. Schmidt and T. Ströhlein, *Relations and Graphs*, EATCS Monographs in Computer Science, Springer-Verlag, Berlin, 1993.

E. Sekerinski, A calculus for predicative programming, in *Second International Conf. on the Mathematics of Program Construction*, Oxford, June 1992 (eds. R. S. Bird, C. C. Morgan and J. C. P. Woodcock), Lecture Notes in Computer Science, 669, Springer-Verlag, 1993, 302–322.

A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific Journal of Mathematics*, **5** (1955), 285–309.

A. Tarski, *On the calculus of relations*, *J. Symbolic Logic*, **6** (1941), 73–89.

F. Tchier and J. Desharnais, A generalisation of a theorem of Mills, in *Proceedings of the Tenth International Symposium on Computer and Information Sciences*, ISCI X, Turkey, October 1995, 27–34.

F. Tchier, *Sémantiques Relationnelles Démoniaques et Vérification de Boucles Non Déterministes*, Ph. D. thesis, Département de mathématiques et de statistique, Université Laval, Canada, 1996. Available from: [http://www2.ift.ulaval.ca/~desharnais/Recherche/Theses/these.Fairouz.pdf](http://www2.ift.ulaval.ca/~desharnais/Recherche/Theses/these.Fairouz.pdf).

F. Tchier and J. Desharnais, Applying a generalization of a theorem of Mills to generalized looping structures, in *Science and Engineering in Software Development. A recognition of Harlan D. Mills’ Legacy*, Los Angeles, CA, May 1999, IEEE Computer Society Press, 1999, 31–38.

F. Tchier, La sémantique démoniaque relationnelle des diagrammes composés, in *Proc. 5th Seminar on Relational Methods in computer Science* (RelMICS’95), Université Laval, Canada, January 2000.

F. Tchier, While loop demonic relational semantics monotype/residual style, *2003 International Conference on Software Engineering Research and Practice (SERP’03)*, Las Vegas, Nevada, USA, June 2003.

F. Tchier, Demonic Semantics: Using monotypes and residuals, *International Journal of Mathematics and Mathematical Sciences, IJMMS*, **3** (2004), 135–160.

F. Tchier, Nondeterministic programming theorem, *WSEAS Trans. On Mathematics*, **5** (2006), 1035–1044.

F. Tchier, *From Operational to Denotational Demonic Semantics of Nondeterministic While Loops*, 10th WSEAS International Conference on Communications and Computers, Athens, Greece, 2006.

F. Tchier, Demonic fixed points, *Acta Cybernetica Journal*, **17** (2006), 533–555.

F. Tchier, Demonic semantics are equal, in *The 2008 International Conference on Foundations of Computer Science (FCS’08)*, Monte Carlo Resort, Las Vegas, Nevada, USA, July, 2008.

M. Walicki and S. Medal, Algebraic approaches to nondeterminism: An overview, *ACM Computing Surveys*, **29** (1997), 30–81.

N. T. E. Ward, *A refinement Calculus for Nondeterministic Expressions*, PhD thesis, University of Queensland, Australia, 1994.

L. Xu, M. Takeichi and H. Iwasaki, Relational semantics for locally nondeterministic programs, *New Generation Computing*, **15** (1997), 339–362.

Received May 2015; revised July 2015.

E-mail address: ftchier@ksu.edu.sa