Concurrent Composition of Objects in Hidden Logic

Dorel Lucanu

TR 03-01, March 2003
Concurrent Composition of Objects in Hidden Logic

Dorel Lucanu
University “A.I.Cuza” of Iași
Faculty of Computer Science
March 11, 2003

Abstract. An operator \( |_\) for concurrent composition of objects specified in hidden logic is proposed and the main properties for this operator are proved.

Keywords: hidden logic, behavioral specification, concurrent connection, composite objects, communicating objects

1 Introduction

Hidden algebra was introduced in [4, 2] to give algebraic semantics for object paradigms. It distinguishes between visible and hidden sorts in the sense that the equality is interpreted strictly on visible sorts and behaviorally on hidden sorts. In hidden algebra the models only behaviorally satisfy specifications, where behaviorally satisfaction means indistinguishability under experiments.

Hidden logic [15, 13, 5] is a generic name for various logics strongly related to hidden algebra and it offers sound rules for behavioral reasoning which can be easily automated.

A simple object is described very naturally in hidden algebra: a hidden sort models the space of local states and the operations can change states (methods) or observe states (observers or attributes). Problems arises when we copy with complex systems whose behavior is given by the concurrent distributed execution of several subsystems which can be also complex. A first solution, based on shared sum and behavioral refinement, was proposed by J. Goguen and R. Diaconescu in the seminal paper [2]. From the categorical point of view, the composite object is a colimit in an appropriate category. Then their method was extended by C. Cărstea for final/cofree families. A method based on projection operators was proposed in [8]. A related approach but where the behavior of the composite object is described by means of events is given in [9].

The approach from this paper combines the idea of colimits from [2] with the theory of structured specifications [1]. The composite object is specified by a structured specification which includes the specifications of the components together with a specification whose the main responsibility is to manage the components. The manager specification is defined by means of an n-ary operator \( |_\) which proved to have the appropriate properties: commutativity, subsystem distinguishability (a slight form of associativity), behavioral equivalence preserving and interleaving property.

2 Specification of objects in hidden logic

A detailed presentation of hidden logic is given in [15, 3]. In this section we present the main definitions corresponding to the particular case of object specification.
Definition 1  An object signature (in hidden logic) consists of:

1. two disjoint partial ordered sets \((V, \leq), (H, \leq)\) called sets of visible and hidden sorts such that \((H, \leq)\) is a rooted tree with the root the maximal element an called the hidden main sort or the state sort;

2. an ordered sorted \((V, \leq) \cup (H, \leq)\)-signature \(\Sigma\) such that any operation \(f \in \Sigma \setminus \Sigma|_V\) is either a method, i.e., its type is of the form \(h v_1 \cdots v_n \rightarrow h\) with \(h \in H\) and \(v_i \in V\) for \(i = 1, \ldots, n\), or an observer (attribute), i.e., its type is of the form \(h v_1 \cdots v_n \rightarrow v\) with \(h \in H\) and \(v \in V\) and \(v_i \in V\) for \(i = 1, \ldots, n\), or a configuration, i.e., its type is of the form \(h_1 \cdots h_k \rightarrow h\) where \(h\) is the main hidden sort and \(h_i < h\);

3. an \(\Sigma|_V\)-algebra \(D\) called data algebra.

Such an object signature is denoted often by \(\Sigma\) and its constituents are denoted by \(H(\Sigma), V(\Sigma), \text{ and } D(\Sigma)\), respectively.

The two restrictions we added to the general definition of hidden signature are: 1) \((H, \leq)\) is a rooted tree and 2) the only operations involving hidden sorts are either methods or observers or configurations. The idea is as follows: a simple object has an unique hidden sort which models the state space of that object. For a concurrent distributed object system, the root of \((H, \leq)\) models the state space of the composite object, which manages the system, and the sons of the root models the state spaces of the component objects. The components can also be complex, i.e., they can have a similar structure. In this way, if \((H', \leq')\) is a subtree of \((H, \leq)\), then the restriction \(\Sigma'\) of \(\Sigma\) to \((H', \leq')\) is a n object signature, too.

Definition 2  Given object signatures \(\Sigma\) and \(\Sigma'\), a object signature morphism \(\Phi : \Sigma \rightarrow \Sigma'\) is a signature morphism \(\Phi = (\psi, \phi)\) such that:

1. \(\psi(v) = v\) for all \(v \in V\);
2. \(\psi(H(\Sigma)) \subseteq H(\Sigma')\) and \(\psi\) preserves the main hidden sort;
3. \(\phi(f) = f\) for each \(f \in \Sigma|_V\);
4. if \(f' \in \Sigma'_{w', s}\) and some sort in \(w'\) lies in \(\psi(H)\), then \(f' = \phi(f)\) for some \(f \in \Sigma\).

We often use the notations \(\Phi(v)\) (\(\Phi(h)\)) for \(\psi(v)\) (\(\psi(h)\)) and \(\Phi(f)\) for \(\psi(f)\).

We denote by \textbf{ObjSig} the category of object signatures. The definition for models remains unchanged:

Definition 3  Given an object signature \(((V, \leq), (H, \leq), \Sigma, D)\), an object \(\Sigma\)-model is a \(\Sigma\)-algebra \(M\) such that \(M|_{\Sigma|_V} = D\). An object \(\Sigma\)-homomorphism \(h : M \rightarrow M'\) is a \(\Sigma\)-homomorphism such that \(h|_{\Sigma|_V} = \text{id}_D\).

Definition 4  Consider given an object signature \(((V, \leq), (H, \leq), \Sigma, D)\) and a subsignature \(\Gamma \subseteq \Sigma\) such that \(\Gamma|_V = \Sigma|_V\). A \(\Gamma\)-context for sort \(s\) is a term in \(T\{\_ : s\}|_\Gamma\) having exactly one occurrence of a special variable \(\_\) of sort \(s\), where \(Z\) is an infinite set of distinct variables. \(C_\Gamma[\_ : s]\) denotes the set of all \(\Gamma\)-contexts for sort \(s\). If \(c \in C_\Gamma[\_ : s]\), then the sort of the term \(c\) is called the result sort of the context \(c\). A \(\Gamma\)-context with visible result sort is called \(\Gamma\)-experiment. If \(c \in C_\Gamma[\_ : s]\) with the result sort \(s'\) and \(t \in T\Sigma(X)_s\), then \(c[t]\) denotes the term in \(T\Sigma(\text{var}(c) \cup X)\) obtained from \(c\) by substituting \(t\) for \(\_\). Furthermore, \(c\) defines a map \([c]_M : M_s \rightarrow [M^{\text{var}}(c) \rightarrow M_s]\) on each object \(\Sigma\)-model \(M\), given by \([c]_M(a)(\vartheta) = a_{\vartheta}(c)\), where \(a_{\vartheta}(c)\) is the variable assignment \(\_ \mapsto a\) \cup \{z \mapsto \vartheta(z) \mid z \in \text{var}(c)\}\). We call \([c]_M\) the interpretation of the context \(c\) in \(M\).
Definition 5 Consider given an object signature \( ((V, \leq), (H, \leq), \Sigma, D) \), a subsignature \( \Gamma \subseteq \Sigma \) such that \( \Gamma |_V = \Sigma |_V \) and an object \( \Sigma \)-model \( M \). The \( \Gamma \)-behavioral equivalence on \( M \), denoted by \( \equiv^\Gamma \), is defined as follows: for any sort \( s \in V \cup H \) and any \( a, a' \in M_s \), \( a \equiv^\Gamma a' \) iff \( [c^M(a)](\vartheta) = [c^M(a')](\vartheta) \) for all \( \Gamma \)-experiments \( c \) and all \( (V \cup H) \)-sorted maps \( \vartheta : \text{var}(c) \to M \).

Given an equivalence \( \sim \) on \( M \), an operation \( f \in \Sigma_{s_1 \ldots s_n, s} \) is congruent for \( \sim \) iff \( [f^M(a_1, \ldots, a_n)] \sim [f^M(a'_1, \ldots, a'_n)] \) whenever \( a_i \sim a'_i \) for \( i = 1, \ldots, n \). An operation \( f \in \Sigma \) is \( \Gamma \)-behaviorally congruent for \( M \) iff it is congruent for \( \equiv^\Gamma \). A hidden \( \Gamma \)-congruence on \( M \) is a \( (V \cup H) \)-equivalence on \( M \) which is the identity on visible sorts and for which each operation in \( \Gamma \) is congruent.

The proof of the following result can be found in [3, 15].

Theorem 6 Consider given an object signature \( ((V, \leq), (H, \leq), \Sigma, D) \), a subsignature \( \Gamma \subseteq \Sigma \) such that \( \Gamma |_V = \Sigma |_V \) and an object \( \Sigma \)-model \( M \). Then \( \Gamma \)-behavioral equivalence is the largest hidden \( \Gamma \)-congruence on \( M \).

Definition 7 An object \( \Sigma \)-model \( M \) \( \Gamma \)-behaviorally satisfies an \( \Sigma \)-equation \( e \) of the form \( (\forall X)t = t' \) iff \( C \) iff for all \( \vartheta : X \to M \), \( \vartheta(t) \equiv^\Gamma \vartheta(t') \) whenever \( \vartheta(u) \equiv^\Gamma \vartheta(v) \) for all \( u = v \) in \( C \). We write \( M \models^\Gamma e \). If \( E \) is a set of \( \Sigma \)-equations, then we write \( M \models^\Gamma E \) iff \( M \models^\Gamma e \) for all \( e \in E \).

Definition 8 An object (behavioral) specification is a triplet \( B = (\Sigma, \Gamma, E) \) consisting of an object \( \Sigma \)-signature, a subsignature \( \Gamma \subseteq \Sigma \) such that \( \Gamma |_V = \Sigma |_V \) and a set \( \Sigma \)-equations. The operations in \( \Gamma \setminus (\Sigma |_V) \) are called behavioral. An object \( \Sigma \)-model \( M \) behaviorally satisfies the specification \( B \) iff \( M \models^\Gamma E \), that is \( M \models^\Gamma E \). We write \( M \models B \) and we also say that \( M \) is a \( B \)-model. We write \( B \models e \) iff \( M \models B \) implies \( M \models e \). An operation \( f \in \Sigma \) is behavioral for \( B \) iff \( f \) is \( \Gamma \)-behaviorally congruent for each \( B \)-model \( M \).

We often denote the constituents of \( B \) by \( \Sigma(B), \Gamma(B) \) and respectively \( E(B) \).

We denote by \[ \text{ObjSpec} \] the category of object specifications.

Example 9 This example is inspired from [12]. An agent is working on an assembly line. He receives jobs on a conveyor belt represented by the port \( i \), and dispatch them after assembly along another conveyor belt represented by \( \pi \). There are three grades of job: easy, neutral and difficult. We are also interested to know how many jobs of each grade are made by the agent.

We start by representing data algebra:

\[
dth \text{DATA-AGENT} \text{ is} \\
\text{pr NAT} \ . \\
sorts \text{JobGrade ErrJobGrade} \ . \\
\text{subsort} \ \text{JobGrade} < \text{ErrJobGrade} \ . \\
\text{ops} \ \text{easy neutral difficult} : \to \text{JobGrade} \ . \\
\text{op none} : \to \text{ErrJobGrade} \ . \\
\text{end} \\
\]

Then we use an object specification to describe the behavior of the agent:

\[
bth \text{AGENT} \text{ is} \\
\text{sort Agent} \ . \\
\text{pr DATA-AGENT} \ . \\
\text{op init} : \to \text{Agent} \ . \\
\text{op startJob} : \text{Agent JobGrade} \to \text{Agent} \ . \\
\text{op endJob} : \text{Agent} \to \text{Agent} \ .
\]
op grade: Agent -> ErrJobGrade.
op #: Agent JobGrade -> Nat.
var A: Agent. vars G G': JobGrade.
eq grade(startJob(A, G)) = G.
eq grade(endJob(A)) = none.
eq #(init, G) = 0.
eq #(endJob(A), G) = #(A, G) + 1 if grade(A) == G.
eq #(endJob(A), G) = #(A, G) if not grade(A) == G.
eq #(startJob(A, G), G') = #(A, G').

AGENT is a simple object specification because Agent is the unique hidden sort and therefore it is the main hidden sort. The operations startJob() and endJob() are methods and the operations grade() and #() are observers (attributes). Here are several examples of behaviorally equivalent states:

- if \( \Gamma = \{\text{startJob()}, \text{endJob()}, \text{grade()}\} \), then
  \[
  \text{startJob(st, G)} \equiv_{\Gamma} \text{startJob(st', G)};
  \]
- if \( \Gamma = \{\text{startJob()}, \text{endJob()}, #()\} \), then
  \[
  \text{startJob(st, G)} \equiv_{\Gamma} \text{startJob(startJob(st, G'), G)};
  \]
- if \( \Gamma = \{\text{startJob()}, \text{endJob()}, #(), \text{grade()}\} \), then
  \[
  \text{startJob(st, G)} \equiv_{\Gamma} \text{startJob(startJob(st, G), G)}.
  \]

3 Concurrent Connection of Objects

In this section we define the concurrent connection operator \(\_ \mid \_\) and we prove its main properties.

Definition 10 Consider \(B_1, \ldots, B_n\) \(n\) object specifications with \(h_i\) the main hidden sort of \(B_i\).
Then \(B = B_1 \mid \cdots \mid B_n\) denotes the least object specification which satisfies:

1. \(B_i \subseteq B\) for all \(i = 1, \ldots, n\);
2. the main hidden sort is a distinct sort denoted by Configuration;
3. \(h_i < \text{Configuration}\) for all \(i = 1, \ldots, n\);
4. if \(k \geq 2\) then for each ordered sequence \((i_1, \ldots, i_k)\), with \(i_j \neq i_{j'}\) if \(j \neq j'\) and \(i_j \in \{1, \ldots, n\}\) for \(j = 1, \ldots, n\), there is in \(B\) an operation
   \[
   \langle \_ , \ldots , \_ \rangle : h_{i_1} \cdots h_{i_k} \rightarrow \text{Configuration}
   \]
5. for each method \(f : h_i w \rightarrow h_i\) in \(B_i\) there are in \(B\) a method \(f : \text{Configuration} w \rightarrow \text{Configuration}\) and an equation of the form:
   \[
   (\forall S_{i_1}, \ldots, S_{i_k}, W)f((S_{i_1}, \ldots, S_{i_k}), W) = (S_{i_1}, \ldots, f(S_{i_j}, W), \ldots, S_{i_k})
   \]
   for each operation \(\langle \_ , \ldots , \_ \rangle : h_{i_1} \cdots h_{i_k} \rightarrow \text{Configuration}\) such that \(i_j = i\) for some \(j\), and an equation of the form:
   \[
   (\forall S_j : h_j, W)f(S_j, W) = S_j
   \]
   for all \(j \neq i\).
6. for each observer (attribute) \( f : h_i \rightarrow v \) in \( B_i \) there are in \( B \) an observer \( f : \text{Configuration} \rightarrow v \) and an equation of the form:

\[
(\forall S_i, \ldots, S_{ik}, W)f(\langle S_i, \ldots, S_{ik}, W \rangle) = f(S_j, W)
\]

for each operation \( \langle \ldots, \ldots, \ldots \rangle : h_i \rightarrow \text{Configuration} \) such that \( i_j = i \) for some \( j \);

7. for each \( i = 1, \ldots, n \) there is in \( B \) an projection operation \( h_i : \text{Configuration} \rightarrow h_i \), an equation \( (\forall S_i : h_i)h_i(S_i) = S_i \), and an equation of the form:

\[
(\forall S_i, \ldots, S_{ij}, \ldots, S_{ik})h_i(\langle S_i, \ldots, S_{ij}, \ldots, S_{ik} \rangle) = S_i
\]

for each operation \( \langle \ldots, \ldots, \ldots \rangle : h_i \rightarrow \text{Configuration} \) such that \( i_j = i \) for some \( j \);

8. if \( B_j \) is itself a concurrent connection and

\[
\langle \ldots, \ldots, \ldots \rangle : h_{j_1} \cdots h_{j_k} \rightarrow h_j
\]

is an operation in \( B_j \) defined by the rule (4), then for each operation

\[
\langle \ldots, \ldots, \ldots \rangle : h_{i_1} \cdots h_{i_k} \rightarrow \text{Configuration}
\]

in \( B \), with \( i_p = j \) for some \( i_p \), there is in \( B \) a new operation

\[
\langle \ldots, \ldots, \ldots \rangle : h_{i_1} \cdots h_{i_p-1} h_{j_1} \cdots h_{j_p} h_{i_p+1} \cdots h_{i_k} \rightarrow \text{Configuration}
\]

and an equation of the form:

\[
(\forall S_{i_1}, \ldots, S_{i_p}, S_{i_p}, S_{i_p+1}, \ldots, S_{i_k}, S_{j_1}, \ldots, S_{j_k})\]

\[
\langle S_{i_1}, \ldots, S_{i_p}, S_{i_p+1}, \ldots, S_{i_k}, S_{j_1}, \ldots, S_{j_k} \rangle = \langle S_{i_1}, \ldots, S_{j_1}, \ldots, S_{j_k} \rangle;
\]

9. \( \Gamma(B) \) includes \( \cup_{i=1}^n \Gamma(B_i) \) and each method and observer \( f \) defined by the rule (5) or (6) which lies in \( \cup_{i=1}^n \Gamma(B_i) \).

We supposed in the definition given above that the methods and observers (attributes) have different names for different specifications; if it does not, then we name each operation \( f \) in \( B_i \) by \( f.B_i \) and we add to composite objects an equation of the form \( (\forall S_i) f.B_i(S_i) = f(S_i) \).

The sort \text{Configuration} models the state space of all possible configurations obtained by the distributed concurrent execution of the components. The relationship \( h_i < \text{Configuration} \) states that configurations with just one active object are possible. The configurations involving more than one objects are modeled by means of the operations \( \langle \ldots, \ldots, \ldots \rangle \) defined by rule (4). The rules (5) and (6) extends the operations of the individual components to arbitrary configurations such that each time the composite object receives a request to execute such an operations, it delegates the corresponding component to do it. The projections operators allow to pick up individual states of the components from the global state of the composite object. The equations defined by the rule (8) realizes a flattening of the configurations for the case of multilevel composite objects. The last rule says that the behavior of the composite object is the sum of the individual behaviors of the components. This is more a convention than a rule because there are practical situations where we wish to “hide” some individual behaviors in order to simplify the behavior of the composite object. Such a case is studied in the section devoted to the communications between objects.

The definition of \( _1 \| _1 \) is a generalization of the operator \( _1 \| \) from BOBJ. The operator \( _1 \| \) proved to be useful in many practical cases but, however, it has a limited power (e.g., \( _1 \| \) is not commutative).
Corollary 11 \( \mathcal{B}_1 | \mathcal{B}_2 \) and \( \mathcal{B}_2 | \mathcal{B}_1 \) are identical.

The behavioral specification of a concurrent connection is a particular case of structured specification [1].

Definition 12 If \( n \) is a nonzero natural number, then \( Dg(n) \) denotes the following diagram:

\[
\begin{array}{c}
1 \\
\vdots \\
2 \\
\bullet \\
n
\end{array}
\]

Definition 13 Given \( n \) object behavioral specifications \( \mathcal{B}_1, \ldots, \mathcal{B}_n \), the concurrent connection of \( \mathcal{B}_1, \ldots, \mathcal{B}_n \) is the structured specification \( \mathcal{B} : Dg(n) \rightarrow \text{ObjSpec} \) given by:

\[
\begin{align*}
\mathcal{B}(i) &= \mathcal{B}_i \text{ for } i = 1, \ldots, n, \\
\mathcal{B}(\bullet) &= \mathcal{B}_1 | \cdots | \mathcal{B}_n, \\
\mathcal{B}(i \to \bullet) &= \mathcal{B}_i \subseteq \mathcal{B}_1 | \cdots | \mathcal{B}_n \text{ for } i = 1, \ldots, n.
\end{align*}
\]

We denote the concurrent connection by \( (\mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{B}_1 | \cdots | \mathcal{B}_n) \) but we often abbreviate this notation to \( \mathcal{B}_1 | \cdots | \mathcal{B}_n \).

Now we can distinguish between simple and complex objects:

Definition 14 A simple object specification is an object specification where the main hidden sort is the only one hidden sort. A composite object specification is a concurrent connection \( \mathcal{B}_1 | \cdots | \mathcal{B}_n \) where \( \mathcal{B}_i \) is a simple or composite object specification.

The definitions for morphisms and models of composite objects follows the line described in [1].

Definition 15 Given two composite object specifications \( \mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{B}_1 | \cdots | \mathcal{B}_n) \) and \( \mathcal{B}' = (\mathcal{B}_1', \ldots, \mathcal{B}_n', \mathcal{B}_1' | \cdots | \mathcal{B}_n'') \), a morphism \( \Phi : \mathcal{B} \rightarrow \mathcal{B}' \) consists of:

1. a functor \( F : Dg(n) \rightarrow Dg(n') \) such that \( F(\bullet) = \bullet' \),
2. a (simple or composite) object specification morphism \( \Phi_i : \mathcal{B}_i \rightarrow \mathcal{B}_F(i) \) for \( i = 1, \ldots, n \), and
3. an object specification morphism \( \Phi : \mathcal{B}_1 | \cdots | \mathcal{B}_n \rightarrow \mathcal{B}_1' | \cdots | \mathcal{B}_n' \) such that \( \Phi|_{\mathcal{B}_i} = \Phi_i \).

Definition 16 Given a composite object specification \( \mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{B}_1 | \cdots | \mathcal{B}_n) \), a \( \mathcal{B} \)-model consists of a \( \mathcal{B}_i \)-model \( M_i \) for \( i = 1, \ldots, n \) and a \( \mathcal{B}_1 | \cdots | \mathcal{B}_n \)-model \( M \) such that \( M|_{\Phi_i} = M_i \) where \( \Phi_i \) denotes the signature inclusion \( \Sigma(\mathcal{B}_i) \subseteq \Sigma(\mathcal{B}_1 | \cdots | \mathcal{B}_n) \).

Example 17 We specify a simple agency consisting of a concurrent connection of two agents:

\[
\begin{align*}
\text{make AG1} & \quad \text{is} \quad \text{AGENT} \ast (\text{sort Agent to Agent1}) \quad \text{end} \\
\text{make AG2} & \quad \text{is} \quad \text{AGENT} \ast (\text{sort Agent to Agent2}) \quad \text{end} \\
\text{make AGENCY} & \quad \text{is} \quad \text{AG1} \mid \text{AG2} \quad \text{end}
\end{align*}
\]

The composite object specification is equivalent to the following structured specification:
bth AGENCY is
pr (AG1 + AG2).

sort Configuration. *** main hidden sort
subsort Agent1 < Configuration.
subsort Agent2 < Configuration.
op <_,_> : Agent1 Agent2 -> Configuration.
op <_,_> : Agent2 Agent1 -> Configuration.

*** AG1’s operations in the new environment
op startJob.AG1 : Configuration JobGrade -> Configuration.
op endJob.AG1 : Configuration -> Configuration.
op grade.AG1 : Configuration -> JobGrade.
op #.AG1 : Configuration JobGrade -> Nat.

*** AG2’s operations in the new environment
op startJob.AG2 : Configuration JobGrade -> Configuration.
op endJob.AG2 : Configuration -> Configuration.
op grade.AG2 : Configuration -> JobGrade.
op #.AG2 : Configuration JobGrade -> Nat.

*** projections
op Agent1 : Configuration -> Agent1.
op Agent2 : Configuration -> Agent2.

var A1 : Agent1. var A2 : Agent2. var G : JobGrade.

*** equations for the new operations
eq startJob.AG1(<A1, A2>, G) = <startJob(A1, G), A2>.
eq endJob.AG1(<A2, A1>, G) = <A2, endJob(A1, G)>.
eq grade.AG1(<A1, A2>) = grade(A1).
eq #.AG1(<A1, A2>, G) = #(A1, G).
eq #.AG1(<A2, A1>, G) = #(A1, G).
eq startJob.AG1(A1, G) = startJob(A1, G).
eq startJob.AG1(A2, G) = A2.
eq endJob.AG1(A1) = endJob(A1).
eq endJob.AG1(A2) = A2.
eq grade.AG1(A1) = grade(A1).
eq #.AG1(A1, G) = #(A1, G).
eq startJob.AG2(<A1, A2>, G) = <A1, startJob(A2, G)>.
eq startJob.AG2(<A2, A1>, G) = <startJob(A2, G), A1>.
eq endJob.AG2(<A1, A2>) = <A1, endJob(A2)>.
eq grade.AG2(<A1, A2>) = grade(A2).
eq grade.AG2(<A2, A1>) = grade(A2).
As we can see above, the concurrent connection of two relatively simple specifications is a specification of a considerable size. This can rise some problems concerning the implementation of this operation. From our experience, systems like BOBJ [15, 5, 14, 6] and Maude [11] are powerful enough to manipulate such large specifications.

The next result shows that the behavioral equivalence for component objects is preserved by the composite object:

**Theorem 18** Consider $B = B_1 \mid \cdots \mid B_n$ with $h_i$ the main hidden sort of $B_i$ for $i = 1, \ldots, n$ and $(M_1, \ldots, M_n, M)$ a $B$-model. If $\langle a_{i_1}, \ldots, a_{i_k} \rangle, \langle a'_{i_1}, \ldots, a'_{i_k} \rangle \in M_{\text{Configuration}}$, then $\langle a_{i_1}, \ldots, a_{i_k} \rangle \equiv \Sigma\Gamma\langle a'_{i_1}, \ldots, a'_{i_k} \rangle$ iff $a_{i_j} \equiv \Sigma\Gamma(h_i) a'_{i_j}$ for all $j = 1, \ldots, k$.

**Proof:** Consider $c[\_ : \text{Configuration}]$ a $\Gamma(B)$-experiment and $\vartheta : \text{var}(c) \rightarrow M$. We check by induction on depth of $c$ that $[c]M(\langle a_{i_1}, \ldots, a_{i_k} \rangle) = [c]_{M_{i_j}}(a_{i_j})$ for certain $j$. If $c = f(\_, t_1, \ldots, t_n)$, then:

$$
[c]M(\langle a_{i_1}, \ldots, a_{i_k} \rangle) = (\text{by definition})
$$

$$
[f]M(\langle a_{i_1}, \ldots, a_{i_k} \rangle, \vartheta(t_1), \ldots, \vartheta(t_n)) = (\text{by rule 10.6})
$$

$$
[f]_{M_{i_j}}(a_{i_j}, \vartheta(t_1), \ldots, \vartheta(t_n)) = (\text{by definition})
$$

$$
[c]_{M_{i_j}}(a_{i_j}).
$$

We suppose now that $c = c'[f(\_, t_1, \ldots, t_n)]$ where $c'$ is a context such that depth$(c') = \text{depth}(c) − 1$. Note that $f$ is necessarily a method. We have:

$$
[c]M(\langle a_{i_1}, \ldots, a_{i_k} \rangle) = (\text{by definition})
$$

$$
[c]'M([f]M(\langle a_{i_1}, \ldots, a_{i_k} \rangle, \vartheta(t_1), \ldots, \vartheta(t_n))) = (\text{by rule 10.5})
$$

$$
[c]'M(\langle a_{i_1}, \ldots, [f]_{M_{i_j}}(a_{i_j}, \vartheta(t_1), \ldots, \vartheta(t_n)) \rangle, \ldots, a_{i_k})) = (\text{by inductive hypothesis})
$$

$$
[c]'_{M_{i_j}}(\langle f \rangle_{M_{i_j}}(a_{i_j}, \vartheta(t_1), \ldots, \vartheta(t_n))) = (\text{by definition})
$$

$$
[c]_{M_{i_j}}(a_{i_j})
$$
Corollary 19 Consider given a composite object specification \( \mathcal{B} = B_1 \mid \cdots \mid B_n \). If \( \langle \_ , \ldots , \_ \rangle \) is of arity \( h_i \cdots h(i_k) \) and \( \pi : \{1,2,\ldots,n\} \rightarrow \{1,2,\ldots,n\} \) is a bijection, then
\[
\mathcal{B} \models (\forall S_1 : h_{i_1}, \ldots, S_{k} : h_{i_k})(S_1, \ldots, S_{k}) = (S_{\pi(1)}, \ldots, S_{\pi(k)})
\]

The following result shows that the composite object has the interleaving property:

Theorem 20 Consider given a composite object specification \( \mathcal{B} = B_1 \mid \cdots \mid B_n \).

1. If \( f_i \) is an observer in \( B_i \) and \( f_j \) is a method in \( B_j \) with \( i \neq j \), then
\[
\mathcal{B} \models (\forall S_{i_1}, \ldots, S_{i_k}, W) f_i(f_j((S_{i_1}, \ldots, S_{i_k}), W)) = f_i((S_{i_1}, \ldots, S_{i_k}), W);
\]

2. If \( f_i \) is a method in \( B_i \) and \( f_j \) is a method in \( B_j \) with \( i \neq j \), then
\[
\mathcal{B} \models (\forall S_{i_1}, \ldots, S_{i_k}, W, W') f_i(f_j((S_{i_1}, \ldots, S_{i_k}), W), W) = f_j(f_i((S_{i_1}, \ldots, S_{i_k}), W), W').
\]

Proof: We prove only the first assertion, the second one being proved in a similar way. It is easy to check that the following equalities hold in \( \mathcal{B} \):
\[
\begin{align*}
f_i(f_j((S_{i_1}, \ldots, S_{i_k}), W'), W) &= \text{ (by rule 10.5)} \\
f_i((\ldots f_j(S_j, W') \ldots), W) &= \text{ (by rule 10.6)} \\
f_i(S_i, W) &= \text{ (by rule 10.6)} \\
f_i((S_{i_1}, \ldots, S_{i_k}), W)
\end{align*}
\]
where we supposed that there are \( p \) and \( q \) such that \( i_p = i \) and \( i_q = j \).

Remark 21 Consider the notations and hypothesis from Theorem 20. If \( f_i \) and \( f_j \) are methods of arities \( h_iw \) and respectively \( h_jw' \), then we can define the operation (method)
\[
f_i \parallel f_j : \text{Configuration}_w w' \rightarrow \text{Configuration}
\]
such that:
\[
\begin{align*}
f_i \parallel f_j((\ldots S_i, \ldots, S_j, \ldots), W, W') &= (\ldots f_i(S_i, W), \ldots, f_j(S_j, W') \ldots) \\
f_i \parallel f_j((\ldots S_j, \ldots, S_i, \ldots), W, W') &= (\ldots f_j(S_j, W'), \ldots, f_i(S_i, W) \ldots)
\end{align*}
\]
We often write \( f_i(S_i, W) \parallel f_j(S_j, W') \) for \( f_i \parallel f_j((\ldots S_i, \ldots, S_j \ldots), W, W') \).

The operator \( _\parallel _\parallel \) is not associative. E.g., the operation
\[
(\_ \_ \_ \_ : h_2h_1h_3 \rightarrow \text{Configuration}(B_1 \mid B_2) \mid B_3)
\]
has no a correspondent in \( B_1 \mid (B_2 \mid B_3) \) and
\[
(\_ \_ \_ \_ : h_1h_3h_2 \rightarrow \text{Configuration} B_1 \mid (B_2 \mid B_3))
\]
has no a correspondent in \( (B_1 \mid B_2) \mid B_3 \). Moreover, this shows that we cannot define a signature morphism from \( B_1 \mid (B_2 \mid B_3) \) to \( (B_1 \mid B_2) \mid B_3 \) or a converse one. However, the two operations are very strong related and this relationship is expressed by the following:
\[
B_1 \mid B_2 \mid B_3 \models (\forall S_1 : h_1, S_2 : h_2, S_3 : h_3)(S_2, S_1, S_3) = (S_1, S_3, S_2)
\]
This support our claim that \( (B_1 \mid B_2) \mid B_3 \) and \( B_1 \mid (B_2 \mid B_3) \) are behavioral bisimilar [10].

The associativity is used often to identify a subexpression into a complex expression. In our terms, this is interpreted as identifying the state of a subsystem into a state of the composite system and we call it subsystem distinguishability property. The following result shows that the operator \( _\parallel _\parallel \) has this property:
Theorem 22 Consider given a composite object specification $\mathcal{B} = \mathcal{B}_1 | \cdots | \mathcal{B}_n$. If $\{i_1 < \cdots < i_k\}$ is a subset of $\{1, 2, \ldots, n\}$, then there is a specification morphism $\Phi : \mathcal{B}_{i_1} | \cdots | \mathcal{B}_{i_k} \to \mathcal{B}$.

Proof: We denote by $\mathcal{B}'$ the left hand side, by $h$ the main hidden sort in $\mathcal{B}$ and by $h'$ the main hidden sort in $\mathcal{B}'$. We define $\Phi : \mathcal{B}' \to \mathcal{B}$ as follows:

1. $\Phi[\mathcal{B}_j]$ is the identity for all $i = 1, \ldots, n + 1$;
2. $\Phi(h') = h$;
3. $\Phi((\_1, \ldots, \_k) : h_{j_1} \cdots h_{j_k} \to h') = (\_1, \ldots, \_k) : h_{j_1} \cdots h_{j_k} \to h$;
4. if $f$ is a method or an observer in $\mathcal{B}_{i_j}$, then $\Phi(f.\mathcal{B}') = f.\mathcal{B}$.

Now it is easy to see that for each equation $e$ in $\mathcal{B}'$ we have $\Phi(e)$ in $\mathcal{B}$.

The next result relates the operators $\_=\_\|\_\|\_\$ (see e.g., [7]) and $\_\|$ and its proof follows direct from definitions:

Proposition 23 Let $\mathcal{B}_1, \ldots, \mathcal{B}_n$ be $n$ object specifications and $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ a bijection. Then there is a specification morphism $\Phi : \mathcal{B}_{\pi(1)}|\cdots|\mathcal{B}_{\pi(n)} \to \mathcal{B}_1|\cdots|\mathcal{B}_n$.

Corollary 24 For each ordered sequence $(i_1, \ldots, i_k)$, with $i_j \neq i_j'$ if $j \neq j'$ and $i_j \in \{1, \ldots, n\}$ for $j = 1, \ldots, n$, there is a object specification morphism $\Phi : \mathcal{B}_{i_1}|\cdots|\mathcal{B}_{i_k} \to \mathcal{B}_1|\cdots|\mathcal{B}_n$.

The concurrent connection is not just a colimit but it has a very close property:

Theorem 25 Consider given a composite object specification $\mathcal{B} = \mathcal{B}_1 | \cdots | \mathcal{B}_n$. If $\mathcal{B}'$ is an object specification such that for each ordered sequence $(i_1, \ldots, i_k)$, with $i_j \neq i_j'$ if $j \neq j'$ and $i_j \in \{1, \ldots, n\}$ for $j = 1, \ldots, n$, there is a specification morphism $\Phi_{i_1 \cdots i_k} : \mathcal{B}_{i_1} \cdots \mathcal{B}_{i_k} \to \mathcal{B}'$, then there is a specification morphism $\Phi : \mathcal{B} \to \text{Der}(\mathcal{B}')$, where $\text{Der}(\mathcal{B}')$ denotes the specification $\mathcal{B}'$ enriched with the derived operations.

Proof: We use the shorter notation $\mathcal{B}_{i_1 \cdots i_k}$ for $\mathcal{B}_{i_1} \cdots \mathcal{B}_{i_k}$. Let $h$ be the main hidden sort of $\mathcal{B}'$. By definition of object specification morphisms, $\Phi_{i_1 \cdots i_k}(\text{Configuration}.\mathcal{B}_{i_1 \cdots i_k}) = h'$.

The morphism $\Phi$ is defined as follows. If $(\_1, \ldots, \_k)$ is a configuration of arity $h_{i_1} \cdots h_{i_k}$, then $\Phi((\_1, \ldots, \_k)) = \Phi_{i_1 \cdots i_k}(\_1, \ldots, \_k)$. If $f$ is an attribute in $\mathcal{B}_i$, then $\Phi(f((S_{i_1}, \ldots, S_{i_k}))) = \Phi_{i_1}(f(S_{i_1}, \ldots, S_{i_k}))$, where $i_j = i$. Similarly, if $f$ is a method in $\mathcal{B}_i$, then $\Phi(f((S_{i_1}, \ldots, S_{i_k})) = \Phi_{i_1 \cdots i_k}(S_{i_1}, \ldots, f(j(S_{i_1}, \ldots, S_{i_k})), \ldots, S_{i_k}))$, where $i_j = i$. If $(\_1, \ldots, \_k)$ is a configuration defined by the rule 10.8, then $\Phi((S_{i_1}, \ldots, S_{j_1}, \ldots, S_{j_1}, \ldots, S_{i_k})) = \Phi_{i_1 \cdots i_k}(S_{i_1}, \ldots, f(\Phi_j((S_{j_1}, \ldots, S_{j_1})), \ldots, S_{i_k}))$. □

4 Communication and Specialization

In this section we show by means of an example how the concurrent connection can be naturally used to describe the communication between objects.

We define first the specification of a memory cell able to store bits:

dth DATA-CELL is
    sort Data .
    ops 0 1 : -> Data .
end
We consider $\Gamma(\text{CELL}) = \{\text{read}, \text{write}\}$. It is easy to see that two cell states are behavioral equivalent iff they store the same data. Then we specify a buffer of capacity two as the concurrent connection of two cells to which we add a method modeling the communication between cells:

```
make CELL1 is CELL * (sort Cell to Cell1) endm
make CELL2 is CELL * (sort Cell to Cell2) endm

bth BUFFER2 is
  pr (CELL1 | CELL2)*(sort Configuration to Buffer2) .
  op comm : Buffer2 -> Buffer2 .
  var C1 : Cell1 . var C2 : Cell2 . var D : Data .
  eq comm(<C1, C2>) = <C1, write(C2, read(C1)> .
  eq comm(<C2, C1>) = <C2, write(C1, read(C2)> .
end
```

The definition of the buffer is very general: we can write in any cell, we can read from any cell, and we can copy the information from a cell into another. If we consider the standard definition for $\Gamma(\text{BUFFER2})$, then we still have:

$$B \equiv B' \iff \text{Cell1}(B) \equiv \text{Cell1}(B') \land \text{Cell2}(B) \equiv \text{Cell2}(B').$$

We specialize now the buffer to behave like a stack (of capacity two):

```
bth STACK2 is
  pr BUFFER2 * (sort Buffer2 to Stack2) .
  op push : Stack2 Data -> Stack2 .
  op pop : Stack2 -> Stack2 .
  op top : Stack2 -> Data .
  var S : Stack2 . var D : Data .
  eq push(S, D) = write.CELL1(comm(Cell1(S), Cell2(S)), D) .
  eq pop(S) = comm(Cell2(S), Cell1(S)) .
  eq top(S) = read.CELL1(S) .
end
```

It is easy to check that the following two equations are satisfied by STACK2:

$$\text{top}(\text{push}(S,D)) = D$$
$$\text{top}(\text{pop}(\text{push}(S,D))) = \text{top}(S).$$
The last equation cannot be generalized because the stack has the capacity bounded by two. If we consider now \( \Gamma(\text{BUFFER2}) = \{\text{push()}, \text{top()}\} \), then we have:

\[
S \equiv_{\Gamma(\text{STACK2})} S' \iff \text{Cell11}(S) \equiv_{\Gamma(\text{CELL})} \text{Cell11}(S).
\]

In other words, we may have \( S \equiv S' \in \text{STAK2} \) and \( \text{Cell2}(S) \not\equiv \text{Cell2}(S') \in \text{CELL} \).

The above example exhibits that the communication between objects can be naturally expressed using the concurrent connection and that the concurrent connection can be easily specialized for doing specific tasks.

## 5 Conclusions and Further Work

The paper presents a fragment of hidden logic suitable for pure object-oriented specifications. The fragment is in fact a subcategory of behavioral specifications which includes simple objects, modeled with a single hidden sort, and composite objects defined using the concurrent connection operator. The definition for the concurrent connection is different from the other approaches we know in present and we claim that it fits better with what it is understood by concurrent connection (composition) in object-oriented theory. Our claim is supported by the properties it satisfies.

In this paper we mainly focused on the definition of the concurrent connection operator and its properties. In the future we intend to investigate the properties of the fragment logic corresponding to object specifications. Then we will try to extend our approach to Behavioral Membership Equational Logic (BMEL) [11]. Many operations we consider in the definition of the concurrent connection are partially defined. We hope that membership assertions will help us to have more accurate definitions for these operations. We also believe that the behavioral bisimulation between objects can be more naturally expressed in BMEL.

### References


