NOTES ON AGENT ALGEBRAS

by

Jerry R. Burch, Roberto Passerone and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M03/38

16 October 2003

NOTES ON AGENT ALGEBRAS

by

Jerry R. Burch, Roberto Passerone and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M03/38

16 October 2003

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Notes on Agent Algebras

Jerry R. Burch[*] Roberto Passerone[†‡] Alberto L. Sangiovanni-Vincentelli[§]

October 16, 2003

Abstract

We introduce Agent Algebra as a general framework that can be used to include a wide variety of models for concurrent systems. We introduce an ordering in the algebra to represent refinement in the model, and study its relationships with respect to the operators of the algebra. In particular, we study the problem of compositionality and provide an extension of the notion of monotonic function to the case of partial functions. We then characterize the order in terms of a substitutability relation that we call conformance. We relate the conformance order for each agent to its maximally compatible agent, called the mirror.

Given models for a plant, a controller and a specification, we often want to determine whether the specification is satisfied by the composition of the plant and the controller. It is also common, given a plant and a specification, to try to characterize all of the controllers that meet the above requirement. We give sufficient conditions for constructing such characterizations in the framework of Agent Algebra, including conditions to be met by the definitions of system composition and system refinement in the models.

[*]Synopsys, Inc. - jrb@synopsys.com
[†]Cadence Design Systems, Inc. - robp@cadence.com
[‡]University of California at Berkeley - roby@eecs.berkeley.edu
[§]University of California at Berkeley - alberto@eecs.berkeley.edu

1 Introduction

This technical report describes some very general methods for constructing different models of concurrent systems, and presents general results on compositional methods and refinement verification. We introduce the notion of an *agent algebra* to formalize the model of a concurrent system. Agent algebra is a broad class of models of computation. They developed out of our work on *concurrency algebra, trace algebra* and *trace structure algebra* [2, 4, 5, 6], which builds on Dill's work on *circuit algebra* and *trace theory* [8]. Through trace structure algebra we have studied concepts, such as *conservative approximations*, that help clarify the relationships between different models of computation. Agent algebra provides a simpler formalism for describing and studying these concepts. The tradeoff is that agent algebra is more abstract and provides less support for constructing models of computation.

An agent algebra (section 3) is a simple abstract algebra with three operations: parallel composition, projection, and renaming. The three operations must satisfy certain axioms that formalize their intuitive interpretation. The domain (or carrier) of an agent algebra is intended to represent a set of processes, or *agents*. Any set can be the domain of an agent algebra if interpretations for parallel composition, projection and renaming that satisfy the axioms can be defined over the set. In this document, whenever we define an interpretation for these three operations, we always show that the interpretation forms an agent algebra, which gives evidence that the interpretation makes intuitive sense.

Agent algebras can be constructed from other agent algebras by the usual devices of direct product, disjoint sum and subalgebra (section 4). We introduce these constructions, and show that they effectively yield new agent algebras. Direct products are useful for example to construct hybrid models and to provide a sort of "signature specification" to a set of agents. The examples are not, however, included in these notes.

In verification and design-by-refinement methodologies a specification is a model of the design that embodies all the possible implementation options. Each implementation of a specification is said to *refine* the specification. In our framework, agent algebras may include a preorder on the agents that represents refinement relationships (section 5). Proving that an implementation refines a specification is often a difficult task. Most techniques decompose the problem into smaller ones that are simpler to handle and that produce the desired result when combined. To make this approach feasible, the operations on the agents must be monotonic with respect to the refinement order. In this document we extend the notion of monotonic function to the case of partial functions, and show under what circumstances compositional verification techniques can be applied.

Expressions can be defined in terms of the operators and the agents of an agent algebra (section 6). We show that under certain conditions every expression can be transformed into an equivalent expression in a particular normal form. The normal form is useful to deal with systems specified as complex interaction of hierarchies, as it allows one to flatten the system to a single parallel composition. The normal form is also important in obtaining closed form solutions to equations involving expressions on agents.

The order of an agent algebra can often be characterized in terms of substitutability as a *conformance* relation (section 7). We introduce the definition of a conformance order by considering the effect of substituting an agent for another agent in every possible context. We parameterize the notion of substitutability using a set of agents, called a *conformance set*. Conformance can be used to verify the order relationship between agents. In these notes we introduce the notion of the *mirror* of an agent (section 8), which, together with a conformance order, reduces the task of refinement verification to computing a parallel composition and checking membership with the conformance set.

Substitutability in every context is often not required. The conformance order and the mirror function is used with an agent algebra to formulate and to solve the problem of synthesizing a local specification subject to a context (section 9). This construction, which is independent of the particular agent algebra considered, is useful in developing synthesis techniques, and can be applied to solve problems such as supervisory control synthesis, engineering changes, rectification and protocol conversion.

These notes include simple examples of agent algebras that deal essentially with the definition of the interface of an agent in terms of its input and output signals. The examples are carried through the document and provide a simple and more practical explanation of the theory developed in these notes. More complex examples that include behavior models are also possible, based for instance on trace structure algebras [2, 4, 5]. The full version of these notes will present these examples and will include a thorough treatment of their conformance order and mirror function.

2 Preliminaries

The algebras we develop in this document have many characteristics in common. This section discusses several of those characteristics.

Each of the algebras has a domain D which contains all of the objects under study for the algebra. We borrow the term "domain" from the progamming language semantics literature; algebraists call D a "carrier".

Associated with each element of D is a set A of *signals*, called an *alphabet*. Signals are used to model communication between elements of D. Typically signals serve as actions and/or state variables that are shared between elements of D, but this need not be the case. Associated with each algebra is a *master alphabet*. The alphabet of each agent must be a subset of the master alphabet. A master alphabet typically plays the role of $\mathcal{A}$ in the following definition.

Definition 2.1. If $\mathcal{A}$ is a set, then A is an *alphabet over* $\mathcal{A}$ iff $A \subseteq \mathcal{A}$.

We often make use of functions that are *over* some domain or master alphabet.

Definition 2.2. Let S be an arbitrary set. A function f of arity n is *over* S iff $dom(f) \subseteq S^n$ and $codom(f) \subseteq S$.

Definition 2.3. Let $\mathcal{A}$ and D be sets. A *renaming operator over master alphabet $\mathcal{A}$ and over domain D* (written *rename*) is a total function such that

1. the domain of *rename* is the set of bijections over $\mathcal{A}$, and
2. the codomain of *rename* is the set of partial functions from D to D.

Definition 2.4. Let $\mathcal{A}$ and D be sets. A *projection operator over master alphabet $\mathcal{A}$ and over domain D* (written *proj*) is a total function such that

1. the domain of *proj* is the set of alphabets over $\mathcal{A}$, and
2. the codomain of *proj* is the set of partial functions from D to D.

Definition 2.5. Let D be a set. A *parallel composition operator over domain D* (written a binary infix operator $\|$) is a partial function over D such that

1. the domain of $\|$ is $D \times D$, and

2. the codomain of $\|$ is D.

The codomain of the operators above are partial functions, and can therefore be undefined for certain arguments. In the rest of this work, we often say that the operator itself is undefined, with the understanding that it is the resulting partial function that really is undefined at a certain argument. In formulas, we use the notation $\downarrow$ to indicate that a function is defined at a particular argument, and $\uparrow$ to indicate that it is undefined.

3 Agent Algebras

Informally, an agent algebra $\mathcal{Q}$ is composed of a domain D which contains the agents under study for the algebra, and of the following operations on agents: parallel composition, projection and renaming. The algebra also includes a master alphabet $\mathcal{A}$, and each agent is characterized by an alphabet A over $\mathcal{A}$. All of this is formalized in the following definitions. Throughout the document, equations are interpreted to imply that the left hand side of the equation is defined iff the right hand side is defined, unless stated otherwise.

Definition 3.1. An *agent algebra* $\mathcal{Q}$ has a domain $\mathcal{Q}.D$ of *agents*, a *master alphabet* $\mathcal{Q}.\mathcal{A}$, and three operators: *renaming* (definition 2.3), *projection* (definition 2.4) and *parallel composition* (definition 2.5), denoted by *rename*, *proj* and $\|$. The function $\mathcal{Q}.\alpha$ associates with each element of D an alphabet A over $\mathcal{A}$. For any p in $\mathcal{Q}.D$, we say that $\mathcal{Q}.\alpha(p)$ is the *alphabet of p*.

The operators of projection, rename and parallel composition must satisfy the axioms given below, where p and p' are elements of D, $A = \alpha(p)$, $A' = \alpha(p')$, B is an alphabet and r is a renaming function.

A1. If $proj(B)(p)$ is defined, then its alphabet is $B \cap A$.

A2. $proj(A)(p) = p$.

A3. If $rename(r)(p)$ is defined, then $A \subseteq dom(r)$ and $\alpha(rename(r)(p)) = r(A)$, where r is naturally extended to sets.

A4. $rename(id_A)(p) = p$.

A5. If $p \parallel p'$ is defined, then its alphabet is $A \cup A'$.

A6. Parallel composition is associative.

A7. Parallel composition is commutative.

The operators have an intuitive correspondence with those of most models of concurrent systems. The operation of renaming corresponds to the instantiation of an agent in a system. Note that since the renaming function is required to be a bijection, renaming is prevented from altering the structure of the agent interface, by for example "connecting" two signals together. Projection corresponds to hiding a set of signals. In fact, the projection operator is here used to *retain* the set of signals that comes as an argument, and hide the remaining signals in the agent. In that sense it corresponds to an operation of scoping. Finally, parallel composition corresponds to the concurrent "execution" of two agents. It is possible to define other operators. We prefer to work with a limited set and add operators only when they can't be derived from existing ones. The three operators presented here are sufficient for the scope of this work.

A1 through A7 formalize the intuitive behavior of the operators and provide some general properties that we want to be true regardless of the model of computation. These properties, together with the ones required for normalization later in section 6, are at the basis of the results of this work.

As described in the above definition, an agent in an agent algebra contains information about what its alphabet is. A simple example of an agent algebra $\mathcal{Q}$ can be constructed by having each agent be nothing more than its alphabet, as follows.

Example 3.2 (Alphabet Algebra). For this example, the master alphabet $\mathcal{Q}.\mathcal{A}$ is an arbitrary set of signal names. The domain $\mathcal{Q}.D$ of the algebra is the set of all subsets of $\mathcal{Q}.\mathcal{A}$. The alphabet of any p in $\mathcal{Q}.D$ is simply p itself. Thus, $\mathcal{Q}.\alpha$ is the identity function. If r is a bijection over $\mathcal{A}$, then $rename(r)(p)$ is defined whenever $p \subseteq dom(r)$, in which case $rename(r)(p)$ is $r(A)$ (where r is naturally extended to sets). If B is a subset of the master alphabet $\mathcal{A}$, then $proj(B)(p)$ is $B \cap p$. Finally, $p \parallel p'$ is $p \cup p'$. It is easy to show that A1 through A7 are satisfied.

On the opposite extreme from the previous example is an agent algebra where all the agents have an empty alphabet. Later, we will show how such an agent algebra can be useful constructing more complex agent algebras in terms of simpler ones.

Example 3.3. This agent algebra can be used to model some quantitative property of an agent, such as maximum power dissipation. The master alphabet $\mathcal{Q}.\mathcal{A}$ is an arbitrary set of signal names. The domain $\mathcal{Q}.D$ of the algebra is the set of non-negative real numbers. For any p in $\mathcal{Q}.D$, the alphabet of p is the empty set. If r is a bijection over $\mathcal{A}$, then $rename(r)(p)$ is p. Similarly, if B is a subset of $\mathcal{A}$, then $proj(B)(p)$ is p. Finally, $p \parallel p'$ is $p + p'$. Again it is easy to show that the axioms are satisfied.

The agent algebra in example 3.3 illustrates a class of agent algebras which we call *nonalphabetic*, since the agents in the algebra have empty alphabets and *rename* and *proj* are identity functions. This class is formally defined as follows.

Definition 3.4. A *nonalphabetic agent algebra* $\mathcal{Q}$ is an agent algebra with the following properties for any p in $\mathcal{Q}.D$:

1. the alphabet of p is the empty set,
2. if r is a bijection over $\mathcal{Q}.\mathcal{A}$, then $rename(r)(p) = p$, and
3. if B is a subset of $\mathcal{Q}.\mathcal{A}$, then $proj(B)(p) = p$.

We can use agent algebras to describe the interface that agents expose to their environment, in terms of the input and output signals. The following definitions provide some examples. For all of the examples, it is straightforward to show that the axioms of agent algebras are satisfied. Also, for all algebras, the master alphabet $\mathcal{Q}.\mathcal{A}$ is an arbitrary set of signal names.

Example 3.5 (IO Agent Algebra). Consider the IO agent algebra $\mathcal{Q}$ defined as follows:

- Agents are of the form $p = (I, O)$ where $I \subseteq \mathcal{Q}.\mathcal{A}$, $O \subseteq \mathcal{Q}.\mathcal{A}$ and $I \cap O = \emptyset$. The alphabet of p is $\alpha(p) = I \cup O$.
- $rename(r)(p)$ is defined whenever $\alpha(p) \subseteq dom(r)$. In that case $rename(r)(p) = (r(I), r(O))$, where r is naturally extended to sets.

- $proj(B)(p)$ is defined whenever $I \subseteq B$. In that case $proj(B)(p) = (I, O \cap B)$.
- $p_1 \| p_2$ is defined whenever $O_1 \cap O_2 = \emptyset$. In that case $p_1 \| p_2 = ((I_1 \cup I_2) - (O_1 \cup O_2), O_1 \cup O_2)$.

For each agent in this algebra we distinguish between the set of the input signals and the set of the output signals. Notice that parallel composition is defined only when the intersection of the output signals of the agents being composed is empty. In other words, for this algebra we require that each signal in the system be controlled by at most one agent. Notice also that it is impossible to hide input signals. This is required to avoid the case where a signal is not part of the interface of an agent, but it is also not controlled by any other agent (similarly to a floating wire).

In [8], Dill defines a slightly different notion of input and output algebra.

Example 3.6 (Dill's IO Agent Algebra). Consider the Dill's IO agent algebra $\mathcal{Q}$ defined as follows:

- Agents are of the form $p = (I, O)$ where $I \subseteq \mathcal{Q}.\mathcal{A}$, $O \subseteq \mathcal{Q}.\mathcal{A}$ and $I \cap O = \emptyset$. The alphabet of p is $\alpha(p) = I \cup O$.
- $rename(r)(p)$ is defined whenever $\alpha(p) = dom(r)$. In that case $rename(r)(p) = (r(I), r(O))$.
- $proj(B)(p)$ is defined whenever $B \subseteq \alpha(p)$ and $I \subseteq B$. In that case $proj(B)(p) = (I, O \cap B)$.
- $p_1 \| p_2$ is defined whenever $O_1 \cap O_2 = \emptyset$. In that case $p_1 \| p_2 = ((I_1 \cup I_2) - (O_1 \cup O_2), O_1 \cup O_2)$.

The definitions are similar to those in example 3.5, except that the operators of renaming and projection are less often defined. When defined, however, the operators coincide with those in example 3.5.

The above two examples are only concerned with the number and the names of the input and output signals. This is appropriate for models that use signals as pure events. Sometimes signals are associated to a set of values. Most models also include the ability to define a *type* for each signal, that restricts the set of possible values that the signal can take. The following example is a formalization of a valued and typed interface that builds upon example 3.5.

Example 3.7 (Typed IO Agent Algebra). In this example we extend the IO agent algebra described in example 3.5 to contain typing information. Let V be a set of values and 2^V be its powerset. The Typed IO agent algebra $\mathcal{Q}$ is defined as follows:

- Agents are of the form $p = f : \mathcal{Q}.\mathcal{A} \to S$ where

$$S = \{c_U\} \cup \{(c_I, v) : v \subseteq 2^V\} \cup \{(c_O, v) : v \subseteq 2^V\}.$$

  where c_U, c_I and c_O are constants that denote unused, input and output signals, respectively. The set v that is associated to an input or an output represents the range of values (i.e. the type) that the signal can assume. The alphabet of p is $\alpha(p) = \{a \in \mathcal{Q}.\mathcal{A} : f(a) \neq c_U\}$. It is also conveniente to defined the set of inputs, outputs and unused signals as follows:

$$\begin{aligned} inputs(p) &= \{a \in \mathcal{Q}.\mathcal{A} : f(a) \in \{c_I\} \times 2^V\} \\ outputs(p) &= \{a \in \mathcal{Q}.\mathcal{A} : f(a) \in \{c_O\} \times 2^V\} \\ unused(p) &= \{a \in \mathcal{Q}.\mathcal{A} : f(a) = c_U\} \end{aligned}$$

  To simplify the notation we denote by $f(a).c$ and $f(a).v$ the components of f.

- *rename*$(r)(p)$ is defined whenever $\alpha(p) \subseteq dom(r)$. When defined, *rename*$(r)(p) = g$ such that for all $a \in \mathcal{Q}.\mathcal{A}$

$$g(a) = \begin{cases} f(r^{-1}(a)) & \text{if } r^{-1}(a) \text{ is defined} \\ c_U & \text{otherwise} \end{cases}$$

- *proj*$(B)(p)$ is defined whenever *inputs*$(p) \subseteq B$. When defined *proj*$(B)(p) = g$ such that for all $a \in \mathcal{Q}.\mathcal{A}$

$$g(a) = \begin{cases} f(a) & \text{if } a \in B \\ c_U & \text{otherwise} \end{cases}$$

- $p \parallel p'$ is defined if
  - *outputs*$(p) \cap$ *outputs*$(p') = \emptyset$;
  - $f(a).v \subseteq f'(a).v$ whenever $f(a).c = c_O$ and $f'(a).c = c_I$.
  - $f'(a).v \subseteq f(a).v$ whenever $f'(a).c = c_O$ and $f(a).c = c_I$.

  When defined, $p \parallel p' = g$ such that for all $a \in \mathcal{Q}.\mathcal{A}$

$$g(a) = \begin{cases} f(a) & \text{if } f(a).c = c_O \text{ and } f'(a).c \neq c_O \\ f'(a) & \text{if } f'(a).c = c_O \text{ and } f(a).c \neq c_O \\ f(a) & \text{if } f'(a).c = c_U \\ f'(a) & \text{if } f(a).c = c_U \\ (c_I, f(a).v \cap f'(a).v) & \text{if } f(a).c = c_I \text{ and } f'(a).c = c_I \end{cases}$$

The definitions are again similar to those in example 3.5. However, the parallel composition operator is restricted to be defined only if the range of values of an output signal is *contained* in the range of values of the corresponding input signal. In addition, if a signal appears as an input in both agents, the range of values for that input in the composition corresponds to the intersection of the original ranges, so that only values consistent with both components can be used when composing with other agents.

4 Construction of Algebras

Several agent algebras can be combined to form a more complex algebra. A simple example of this is the product of two algebras.

Definition 4.1 (Product). Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras with the same master alphabet (i.e., $\mathcal{Q}_1.\mathcal{A} = \mathcal{Q}_2.\mathcal{A}$). The product of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (written $\mathcal{Q}_1 \times \mathcal{Q}_2$) is the agent algebra $\mathcal{Q}$ such that

1. $\mathcal{Q}.\mathcal{A} = \mathcal{Q}_1.\mathcal{A}$,
2. $\mathcal{Q}.D = \mathcal{Q}_1.D \times \mathcal{Q}_2.D$,
3. $\alpha(\langle p_1, p_2 \rangle) = \alpha(p_1) \cup \alpha(p_2)$,
4. *rename*$(r)(\langle p_1, p_2 \rangle) = \langle$*rename*$(r)(p_1)$, *rename*$(r)(p_2)\rangle$ if both *rename*$(r)(p_1)$ and *rename*$(r)(p_2)$ are defined, otherwise it is undefined,

5. $proj(B)(\langle p_1, p_2\rangle) = \langle proj(B)(p_1), proj(B)(p_2)\rangle$ if both $proj(B)(p_1)$ and $proj(B)(p_2)$ are defined, otherwise it is undefined,

6. $\langle p_1, p_2\rangle \parallel \langle p'_1, p'_2\rangle = \langle p_1 \parallel p'_1, p_2 \parallel p'_2\rangle$ if both $p_1 \parallel p'_1$ and $p_2 \parallel p'_2$ are defined, otherwise it is undefined.

In the product, each agent is a pair of agents, each from one of the original algebras. The operators are defined component-wise. It is easy to prove that the product of two agent algebras is again an agent algebra.

Theorem 4.2. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras, and let $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ be their product. Then $\mathcal{Q}$ is an agent algebra.

Proof: To prove the validity of the axioms simply apply the definitions and the basic commutative, distributive and associative properties of the operations involved. □

Products of algebras are useful to combine in one single model the information contained in two different models.

Example 4.3. Recall the agent algebra examples in example 3.2 and example 3.3, which have domains of $2^{\mathcal{A}}$ and the non-negative real numbers, respectively. The cross product of these two agent algebras combines the information of the two individual algebras.

A second example of construction is the disjoint sum of two algebras.

Definition 4.4 (Disjoint Sum). Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras with master alphabet $\mathcal{A}_1 = \mathcal{Q}_1.\mathcal{A}$ and $\mathcal{A}_2 = \mathcal{Q}_2.\mathcal{A}$, respectively. The disjoint sum of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (written $\mathcal{Q}_1 \uplus \mathcal{Q}_2$) is the agent algebra $\mathcal{Q}$ such that

1. $\mathcal{Q}.\mathcal{A} = \mathcal{Q}_1.\mathcal{A} \uplus \mathcal{Q}_2.\mathcal{A}$,

2. $\mathcal{Q}.D = \mathcal{Q}_1.D \uplus \mathcal{Q}_2.D$,

3. $\alpha(p) = \begin{cases} \mathcal{Q}_1.\alpha(p) & \text{if } p \in \mathcal{Q}_1.D \\ \mathcal{Q}_2.\alpha(p) & \text{if } p \in \mathcal{Q}_2.D \end{cases}$

4. $rename(r)(p) = \begin{cases} \mathcal{Q}_1.rename(r)(p) & \text{if } p \in \mathcal{Q}_1.D \\ \mathcal{Q}_2.rename(r)(p) & \text{if } p \in \mathcal{Q}_2.D \end{cases}$

5. $proj(B)(p) = \begin{cases} \mathcal{Q}_1.proj(B)(p) & \text{if } p \in \mathcal{Q}_1.D \\ \mathcal{Q}_2.proj(B)(p) & \text{if } p \in \mathcal{Q}_2.D \end{cases}$

6. $p \parallel p' = \begin{cases} p\mathcal{Q}_1. \parallel p' & \text{if both } p \in \mathcal{Q}_1.D \text{ and } p' \in \mathcal{Q}_1.D \\ p\mathcal{Q}_2. \parallel p' & \text{if both } p \in \mathcal{Q}_2.D \text{ and } p' \in \mathcal{Q}_2.D \\ \text{undefined} & \text{otherwise} \end{cases}$

In a disjoint sum, two algebras are placed side by side in the same algebra. The agents of each algebra, however, have no interaction with the agents of the other algebra. For this reason the rest of this work will concentrate on products of algebras. Nonetheless, it is easy to show that the disjoint sum of agent algebras is again an agent algebra.

Theorem 4.5. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras, and let $\mathcal{Q} = \mathcal{Q}_1 \uplus \mathcal{Q}_2$ be their disjoint sum. Then $\mathcal{Q}$ is an agent algebra.

If $\mathcal{Q}'$ is an agent algebra and $D \subseteq D'$ is a subset of the agents that is closed in D' under the application of the operators, then D can be used as the domain of a subalgebra $\mathcal{Q}$ of $\mathcal{Q}'$.

Definition 4.6 (Subalgebra). Let $\mathcal{Q}$ and $\mathcal{Q}'$ be agent algebras over the same master alphabet $\mathcal{A}$. Then $\mathcal{Q}$ is called a *subalgebra* of $\mathcal{Q}'$, written $\mathcal{Q} \subseteq \mathcal{Q}'$, if and only if

1. $\mathcal{Q}.D \subseteq \mathcal{Q}'.D$
2. The operators of projection, renaming and parallel composition in $\mathcal{Q}$ are the restrictions to $\mathcal{Q}.D$ of the operators of $\mathcal{Q}'$.

Clearly, the above definition implies that $\mathcal{Q}.D$ is closed in $\mathcal{Q}'.D$ under the application of the operations of agent algebra. Conversely, every subset of $\mathcal{Q}'.D$ that is closed under the application of the operations is the domain of a subalgebra $\mathcal{Q}$ when the operators are the restriction to the subset of the corresponding operators in $\mathcal{Q}'$. The reason why $\mathcal{Q}$ is an agent algebra in that case follows from the fact that the axioms are valid in the substructure, since A1 to A7 are true of all agents in the superalgebra, and therefore must be true of all agents in the subalgebra. The following is an interesting example of this fact.

Theorem 4.7. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras, and let $\mathcal{Q}' = \mathcal{Q}_1 \times \mathcal{Q}_2$ be their cross product. Consider the subset $S \subseteq \mathcal{Q}'.D$ such that for all agents $\langle p_1, p_2 \rangle \in S$, $\alpha(p_1) = \alpha(p_2)$. Then S is closed in $\mathcal{Q}'.D$ under the operations of projection, renaming and parallel composition.

Proof: Let $p = \langle p_1, p_2 \rangle$ and $q = \langle q_1, q_2 \rangle$ be elements of S. The proof is composed of the following cases.

- If $proj(B)(p)$ is defined then

$$
\begin{aligned}
& p \in S \\
& \quad \Leftrightarrow \quad \alpha(p_1) = \alpha(p_2) \\
& \quad \text{by A1} \\
& \quad \Rightarrow \quad \alpha(proj(B)(p_1)) = \alpha(proj(B)(p_2)) \\
& \quad \Rightarrow \quad proj(B)(p) = \langle proj(B)(p_1), proj(B)(p_2) \rangle \in S
\end{aligned}
$$

- If $rename(r)(p)$ is defined then

$$
\begin{aligned}
& p \in S \\
& \quad \Leftrightarrow \quad \alpha(p_1) = \alpha(p_2) \\
& \quad \text{by A3} \\
& \quad \Rightarrow \quad \alpha(rename(r)(p_1)) = \alpha(rename(r)(p_2)) \\
& \quad \Rightarrow \quad rename(r)(p) = \langle rename(r)(p_1), rename(r)(p_2) \rangle \in S
\end{aligned}
$$

- If $p \parallel q$ is defined then

$$
\begin{aligned}
& p \in S \wedge q \in S \\
& \quad \Leftrightarrow \quad \alpha(p_1) = \alpha(p_2) \wedge \alpha(q_1) = \alpha(q_2) \\
& \quad \text{by A5} \\
& \quad \Rightarrow \quad \alpha(p_1 \parallel q_1) = \alpha(p_2 \parallel q_2) \\
& \quad \Rightarrow \quad p \parallel q = \langle p_1 \parallel q_1, p_2 \parallel q_2 \rangle \in S
\end{aligned}
$$

□

Since S is closed, the algebra $\mathcal{Q}$ that has the set S (the agent pairs that have the same alphabet) as the domain, and the restriction to S of the operators of $\mathcal{Q}'$, is a subalgebra of $\mathcal{Q}_1 \times \mathcal{Q}_2$.

5 Ordered Agent Algebras

To study the concepts of refinement and conservative approximations, we can add a preorder or a partial order to an agent algebra. The result is called a *preordered agent algebra* or a *partially ordered agent algebra*, respectively.

We require that the functions in an ordered agent algebra be monotonic relative to the ordering. However, since these are partial functions, this requires generalizing monotonicity to partial functions. The following definition gives two different generalizations. Later we discuss which of these best suits our needs.

Definition 5.1. Let D_1 and D_2 be preordered sets. Let f be a partial function from D_1 to D_2. Let

$$
\begin{aligned}
D_2^\top &= D_2 \cup \{\top\} \\
D_2^\perp &= D_2 \cup \{\perp\},
\end{aligned}
$$

where $\top$ and $\perp$ are not elements of D_2. The preorders over $D_2^\top$ and $D_2^\perp$ are the extensions of the preorder over D_2 such that

$$p_2 \preceq \top \wedge \top \not\preceq p_2$$

and

$$p_2 \not\preceq \perp \wedge \perp \preceq p_2,$$

respectively, for every p_2 in D_2. Let $f_\top$ and $f_\perp$ be the total functions from D_1 to $D_2^\top$ and $D_2^\perp$, respectively, such that for all p_1 in D_1

$$
\begin{aligned}
f_\top(p_1) &= \begin{cases} f(p_1), & \text{if } f(p_1) \text{ is defined;} \\ \top, & \text{otherwise;} \end{cases} \\
f_\perp(p_1) &= \begin{cases} f(p_1), & \text{if } f(p_1) \text{ is defined;} \\ \perp, & \text{otherwise.} \end{cases}
\end{aligned}
$$

We say the function f is $\top$-*monotonic* iff $f_\top$ is monotonic. Analogously, the function f is $\perp$-*monotonic* iff $f_\perp$ is monotonic.

Recall that the formula $p \preceq p'$ intuitively means that p can be substituted for p' in any context. If p is an undefined expression (such as might result from applying a partial function), intuitively it cannot not be substituted for any other agent (except another undefined expression). Thus, an undefined expression should be treated as a maximal element relative to the ordering. Therefore, we require that functions in ordered agent algebras be $\top$-monotonic.

Definition 5.2. A *preordered (partially ordered) agent algebra* is an agent algebra $\mathcal{Q}$ with a preorder (partial order) $\mathcal{Q}.\preceq$ such that for all alphabets B over $\mathcal{Q}.\mathcal{A}$ and all bijections r over $\mathcal{Q}.D$, the partial functions $\mathcal{Q}.rename(r)$, $\mathcal{Q}.proj(B)$ and $\mathcal{Q}.\|$ are $\top$-monotonic. The preorder (partial order) $\mathcal{Q}.\preceq$ is called the *agent order* of $\mathcal{Q}$.

Definition 5.3. Let $\mathcal{Q}$ be a preordered agent algebra. We define the relation "$\approx$" to be the equivalence relation induced by the preorder "$\preceq$". That is

$$p \approx q \Leftrightarrow p \preceq q \wedge q \preceq p.$$

Corollary 5.4. If $\mathcal{Q}$ be a partially ordered agent algebra, then

$$p \approx q \Leftrightarrow p = q.$$

The parallel composition operator is the basis of compositional methods for both design and verification. Monotonicity is required for these methods to work correctly. Henzinger et al. [7] propose to distinguish between *interface* and *component* algebras. Corollary 5.6 below shows that because parallel composition is $\top$-monotonic in an ordered agent algebra, it supports an inference rule identical to the "compositional design" rule for interface algebras. Similarly, component algebras have a "compositional verification" rule that corresponds to $\bot$-monotonic functions. This suggests that the ordering of a component algebra cannot be interpreted as indicating substitutability.

Theorem 5.5. Let f be a $\top$-monotonic partial function. If $p \preceq p'$ and $f(p')$ is defined, then $f(p)$ is defined and $f(p) \preceq f(p')$.

Proof: Let $f_\top$ be as described in definition 5.1. Assume $p \preceq p'$ and $f(p')$ is defined. To show by contradiction that $f(p)$ is defined, start by assuming otherwise. Then, $f_\top(p)$ is equal to $\top$ and $f_\top(p')$ is not. This leads to a contradiction since $p \preceq p'$ and $f_\top$ is monotonic. Also, since $f(p)$ and $f(p')$ are defined, it follows easily from the monotonicity of $f_\top$ that $f(p) \preceq f(p')$. □

Corollary 5.6. Let $\|$ be the composition function of a preordered agent algebra. If $p_1 \preceq p_1'$, $p_2 \preceq p_2'$ and $p_1' \| p_2'$ is defined, then $p_1 \| p_2$ is defined and $p_1 \| p_2 \preceq p_1' \| p_2'$.

Proof: Since $\|$ is $\top$-monotonic by the definition of a preordered agent algebra (definition 5.2), this is simply specializing theorem 5.5 to a binary function. □

The rest of this section is devoted to examples. For each example we derive necessary conditions that the order must satisfy in order for the operators to be $\top$-monotonic. We then choose a particular order, and show that the operators are in fact $\top$-monotonic relative to the order.

Example 5.7 (Alphabet Algebra). Consider the alphabet agent algebra described in example 3.2. The condition of $\top$-monotonicity imposes restrictions on the kind of orders that can be employed in the algebra. In this particular case, the order must be such that $p \preceq p'$ only if $p \subseteq p'$.

Theorem 5.8. Let $\preceq$ be an order for $\mathcal{Q}$ such that *rename*, *proj* and $\|$ are $\top$-monotonic. Then $p \preceq p'$ only if $p \subseteq p'$.

Proof: Let $p \preceq p'$ and let r be a renaming function such that $p' = dom(r)$. Then $rename(r)(p')$ is defined. Since *rename* is $\top$-monotonic, also $rename(r)(p)$ is defined. Therefore $p \subseteq dom(r) = p'$. □

The above result only provides a necessary condition on the order so that the operators are $\top$-monotonic. Any particular choice of order must still be shown to make the operators $\top$-monotonic. Consider, for instance, the order $\preceq$ that corresponds exactly to $\subseteq$, that is $p \preceq p'$ if and only if $p \subseteq p'$. Then

Theorem 5.9. The operators *rename*, *proj* and $\|$ are $\top$-monotonic with respect to $\subseteq$.

Proof: Let $p \subseteq p'$.

- Assume $rename(r)(p')$ is defined. Then $p' \subseteq dom(r)$. Thus, since $p \subseteq p'$, also $p \subseteq dom(r)$, so that $rename(r)(p)$ is defined. In addition since r is a bijection and $p \subseteq p'$

$$rename(r)(p) = r(p) \subseteq r(p') = rename(r)(p')$$

- Let B be a subset of $\mathcal{A}$. Then $proj(B)(p')$ and $proj(B)(p)$ are both defined. In addition, since $p \subseteq p'$,

$$proj(B)(p) = p \cap B \subseteq p' \cap B = proj(B)(p').$$

- Let q be an agent. Then $p' \| q$ and $p \| q$ are both defined. In addition since $p \subseteq p'$

$$p \| q = p \cup q \subseteq p' \cup q = p' \| q.$$

□

Example 5.10 (IO Agent Algebra). Consider the IO agent algebra $\mathcal{Q}$ defined in example 3.5. The requirement that the functions be $\top$-monotonic places a corresponding requirement on the order that can be defined in the algebra.

Theorem 5.11. Let $\preceq$ be an order for $\mathcal{Q}$ such that *rename*, *proj* and $\|$ are $\top$-monotonic. Then $p \preceq p'$ only if $I \subseteq I'$ and $O = O'$.

Proof: Let $p \preceq p'$.

- We first prove that $I \subseteq I'$. Since $I' \subseteq I'$, then $proj(I')(p')$ is defined. Since *proj* is $\top$-monotonic, then also $proj(I')(p)$ must be defined. Therefore it must be $I \subseteq I'$.

- We now prove that $O \subseteq O'$. Assume, by contradiction, that there exists $o \in O$ such that $o \notin O'$. Consider $q = (O', I' \cup \{o\})$. Then $p' \| q$ is defined because $O' \cap (I' \cup \{o\}) = \emptyset$ since by hypothesis $O' \cap I' = \emptyset$ and $o \notin O'$. Since $\|$ is $\top$-monotonic then also $p \| q$ must be defined. But then it must be $O \cap (I' \cup \{o\}) = \emptyset$, which implies $o \notin O$, a contradiction. Hence $O \subseteq O'$.

- Finally we prove that $O' \subseteq O$. Consider the agent $q = (O', I')$. Since by definition $O' \cap I' = \emptyset$, then $p' \parallel q$ is defined and

$$p' \parallel q = ((I' \cup O') - (O' \cup I'), O' \cup I') = (\emptyset, O' \cup I').$$

Since $\parallel$ is T-monotonic, then also $p \parallel q$ is defined and

$$p \parallel q = ((I \cup O') - (O \cup I'), O \cup I').$$

Since $\parallel$ is T-monotonic it must be $p \parallel q \preceq p' \parallel q$. Since *proj* is T-monotonic, it must be

$$\begin{array}{l} (I \cup O') - (O \cup I') \subseteq \emptyset \\ (I \cup O') - (O \cup I') = \emptyset \\ (I \cup O') \subseteq (O \cup I') \\ \quad \text{Since } I \subseteq I' \\ O' \subseteq (O \cup I') \\ \quad \text{Since } O' \cap I' = \emptyset \\ O' \subseteq O. \end{array}$$

□

The converse is not true. That is, it is not the case that if $\preceq$ is an order for $\mathcal{Q}$ such that $p \preceq p'$ only if $I \subseteq I'$ and $O = O'$, then the operators *rename*, *proj* and $\parallel$ are T-monotonic. For example, assume $rename(r)(p')$ is defined. Then we can show that $rename(r)(p)$ is defined. However, since we don't have sufficient conditions for the ordering, the hypothesis are insufficient to show that $rename(r)(p) \preceq rename(r)(p')$. Similarly for the other functions in the algebra.

For the purpose of this example we choose the order $\preceq$ so that $p \preceq p'$ if and only if $I \subseteq I'$ and $O = O'$.

Theorem 5.12. The functions *rename*, *proj* and $\parallel$ are T-monotonic with respect to $\preceq$.

Proof: Let $p \preceq p'$.

- Assume $rename(r)(p')$ is defined. Then $A' \subseteq dom(r)$. By hypothesis, $A \subseteq A'$, so that $A \subseteq dom(r)$. Therefore $rename(r)(p)$ is defined. Since r is a bijection

$$\begin{array}{rcl} I \subseteq I' & \Rightarrow & r(I) \subseteq r(I') \\ O = O' & \Rightarrow & r(O) = r(O') \end{array}$$

Hence $rename(r)(p) \preceq rename(r)(p')$.

- Assume $proj(B)(p')$ is defined. Then $I' \subseteq B$. By hypothesis, $I \subseteq I'$, so that $I \subseteq B$. Therefore $proj(B)(p)$ is defined. In addition

$$\begin{array}{rcl} I \subseteq I' & \Rightarrow & I \subseteq I' \\ O = O' & \Rightarrow & O \cap B = O' \cap B. \end{array}$$

Hence $proj(B)(p) \preceq proj(B)(p')$.

- Assume $p' \parallel q$ is defined, where $q = (I_q, O_q)$. Then $O' \cap O_q = \emptyset$. By hypothesis, $O = O'$ so that $O \cap O_q = \emptyset$. Therefore $p \parallel q$ is defined. In addition

$$\begin{aligned} p' \parallel q &= ((I' \cup I_q) - (O' \cup O_q), O' \cup O_q) \\ p \parallel q &= ((I \cup I_q) - (O \cup O_q), O \cup O_q) \end{aligned}$$

Clearly since $O = O'$

$$O \cup O_q = O' \cup O_q.$$

Therefore, since $I \subseteq I'$

$$(I \cup I_q) - (O' \cup O_q) \subseteq (I' \cup I_q) - (O' \cup O_q).$$

Hence $p \parallel q \preceq p' \parallel q$.

□

Example 5.13 (Dill's IO Agent Algebra). Consider now the Dill style IO agents described in example 3.6. Because the *rename* operator has a further restriction that the domain of the renaming function r be equal to the alphabet of the agent being renamed, the order that results in T-monotonic function is completely determined. In particular

Theorem 5.14. Let $\preceq$ be an order for $\mathcal{Q}$. Then *rename*, *proj* and $\parallel$ are T-monotonic with respect to $\preceq$ if and only if for all agents p and p', $p \preceq p'$ if and only if $p = p'$.

Proof: For the forward direction, assume $\preceq$ is an order such that the functions are T-monotonic. Let $p = (I, O)$ and $p' = (I', O')$ be two agents. Clearly, if $p = p'$, then $p \preceq p'$, since $\preceq$ is reflexive. Conversely, assume $p \preceq p'$.

We first show that $A = A'$. Since $dom(id_{A'}) = A' = \alpha(p')$, then $rename(id_{A'})(p')$ is defined. Since *rename* is T-monotonic, then also $rename(id_{A'})(p)$ is defined. Thus $A = \alpha(p) = dom(id_{A'}) = A'$.

We then show that $I \subseteq I'$. Since $I' \subseteq A'$ and $I' \subseteq I'$, then $proj(I')(p')$ is defined. Since *proj* is T-monotonic, then also $proj(I')(p)$ is defined. Thus $I' \subseteq A$ and $I \subseteq I'$.

Finally we show that $I = I'$ and $O = O'$. Since $O' \cap I' = \emptyset$, then $p' \parallel (O', I')$ is defined. Since $\parallel$ is T-monotonic, then also $p \parallel (O', I')$ is defined. Thus $O \cap I' = \emptyset$. But since, by the above arguments, $O \cup I = O' \cup I'$ then $I' \subseteq I$, and thus $I = I'$. Therefore it must also be $O = O'$.

For the reverse direction, we note than any function is T-monotonic on the discrete order. □

Thus for this example we must choose the order such that $p \preceq p'$ if and only if $I = I'$ and $O = O'$.

Example 5.15 (Typed IO Agent Algebra). Consider the Typed IO agent algebra $\mathcal{Q}$ defined in example 3.7. As for IO agents, T-monotonicity restricts the set of orders that can be applied to the algebra.

Theorem 5.16. Let $\preceq$ be an order for $\mathcal{Q}$ such that *rename*, *proj* and $\parallel$ are T-monotonic. Then $p \preceq p'$ only if $inputs(p) \subseteq inputs(p')$ and $outputs(p) = outputs(p')$, and for all $a \in \mathcal{Q}.\mathcal{A}$, if $f(a).c = c_I$ then $f(a).v \supseteq f'(a).v$, and if $f(a).c = c_O$ then $f(a).v \subseteq f'(a).v$.

Proof: It is easy to adapt the proof of theorem 5.11 to show that $p \preceq p'$ only if

$$\begin{aligned} \mathit{inputs}(p) &\subseteq \mathit{inputs}(p') \\ \mathit{outputs}(p) &= \mathit{outputs}(p') \end{aligned}$$

To prove the rest of the theorem, let $p \preceq p'$ and let $q = f_q$ be the agent such that for all $a \in \mathcal{Q}.\mathcal{A}$

$$f_q(a) = \begin{cases} (c_O, v) & \text{if } f'(a) = (c_I, v) \\ (c_I, v) & \text{if } f'(a) = (c_O, v) \\ c_U & \text{otherwise} \end{cases}$$

so that $\mathit{inputs}(q) = \mathit{outputs}(p')$ and $\mathit{outputs}(q) = \mathit{inputs}(p')$. Then clearly $p' \parallel q$ is defined. Since $\parallel$ is T-monotonic, $p \parallel q$ must also be defined. In fact, since $\mathit{outputs}(p) = \mathit{outputs}(p')$ we already know that $\mathit{outputs}(p) \cap \mathit{outputs}(q) = \emptyset$. Assume now that $a \in \mathcal{Q}.\mathcal{A}$ and $f(a).c = c_I$. Then also $f'(a).c = c_I$, and $f_q(a).c = c_O$. Hence, since $p \parallel q$ is defined, $f_q(a).v \subseteq f(a).v$. But $f_q(a).v = f'(a).v$, thus $f(a).v \supseteq f'(a).v$.

Similarly, assume that $a \in \mathcal{Q}.\mathcal{A}$ and $f(a).c = c_O$. Then also $f'(a).c = c_O$, and $f_q(a).c = c_I$. Hence, since $p \parallel q$ is defined, $f(a).v \subseteq f_q(a).v$. But $f_q(a).v = f'(a).v$, thus $f(a).v \subseteq f'(a).v$. □

Given this result, we choose to order the Typed IO agents so that $p \preceq p'$ if and only if $\mathit{inputs}(p) \subseteq \mathit{inputs}(p')$ and $\mathit{outputs}(p) = \mathit{outputs}(p')$, and for all $a \in \mathcal{Q}.\mathcal{A}$, if $a \in \mathit{inputs}(p)$ then $f(a).v \supseteq f'(a).v$, and if $a \in \mathit{outputs}(p)$ then $f(a).v \subseteq f'(a).v$.

Theorem 5.17. The operations of *rename*, *proj* and $\parallel$ are T-monotonic with respect to $\preceq$.

Proof: The proof of this theorem is similar to the proof of theorem 5.12 and is left as an exercise. □

5.1 Construction of Algebras

In section 4 we have introduced several constructions used to create new algebras from existing ones. In this section we extend those constructions to include the agent order.

The order in the product is the usual point-wise extension.

Definition 5.18 (Product - Order). Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be ordered agent algebras with the same master alphabet. The product of $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ is defined as in definition 4.1 with the order such that

$$\langle p_1, p_2 \rangle \preceq \langle p_1', p_2' \rangle \Leftrightarrow p_1 \preceq p_1' \wedge p_2 \preceq p_2'.$$

Theorem 5.19. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be ordered agent algebras, and let $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ be their product. Then $\mathcal{Q}$ is an ordered agent algebra.

Proof: We must show that the operators are T-monotonic. Here we only show the case for projection, since the other cases are similar.

Let $\langle p_1, p_2 \rangle \preceq \langle p_1', p_2' \rangle$, and assume $\mathit{proj}(B)(\langle p_1', p_2' \rangle)$ is defined. Then both $\mathit{proj}(B)(p_1')$ and $\mathit{proj}(B)(p_2')$ are defined. By definition 5.18, since $\langle p_1, p_2 \rangle \preceq \langle p_1', p_2' \rangle$, also $p_1 \preceq p_1'$ and $p_2 \preceq p_2'$. Thus, since *proj* is T-monotonic in $\mathcal{Q}_1$ and $\mathcal{Q}_2$, $\mathit{proj}(B)(p_1)$ and $\mathit{proj}(B)(p_2)$ are defined, and

$$\mathit{proj}(B)(p_1) \preceq \mathit{proj}(B)(p_1') \wedge \mathit{proj}(B)(p_2) \preceq \mathit{proj}(B)(p_2').$$

Therefore, by definition 5.18, also $proj(B)(\langle p_1, p_2\rangle)$ is defined and

$$proj(B)(\langle p_1, p_2\rangle) \preceq proj(B)(\langle p'_1, p'_2\rangle).$$

Hence $proj$ is $\top$-monotonic in $\mathcal{Q}$. □

The order in the disjoint sum corresponds to the orders in the components.

Definition 5.20 (Disjoint Sum - Order). Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be ordered agent algebras. The disjoint sum of $\mathcal{Q} = \mathcal{Q}_1 \uplus \mathcal{Q}_2$ is defined as in definition 4.4 with the order such that $p \preceq p'$ if and only if either

$$p \in \mathcal{Q}_1.D \wedge p' \in \mathcal{Q}_1.D \wedge p \preceq_{\mathcal{Q}_1} p',$$

or

$$p \in \mathcal{Q}_2.D \wedge p' \in \mathcal{Q}_2.D \wedge p \preceq_{\mathcal{Q}_2} p'.$$

Theorem 5.21. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be ordered agent algebras, and let $\mathcal{Q} = \mathcal{Q}_1 \uplus \mathcal{Q}_2$ be their product. Then $\mathcal{Q}$ is an ordered agent algebra.

The order in the subalgebra must correspond exactly to the order in the superalgebra.

Definition 5.22 (Subalgebra - Order). Let $\mathcal{Q}'$ be an ordered agent algebra. The agent algebra $\mathcal{Q}$ is a subalgebra of $\mathcal{Q}'$ if and only if

- $\mathcal{Q}$ is a subalgebra of $\mathcal{Q}'$ (definition 4.6), and
- for all $p, p' \in \mathcal{Q}.D$, $p \preceq_{\mathcal{Q}} p'$ if and only if $p \preceq_{\mathcal{Q}'} p'$.

Theorem 5.23. Let $\mathcal{Q}'$ be an ordered agent algebra and let $\mathcal{Q} \subseteq \mathcal{Q}'$. Then $\mathcal{Q}$ is an ordered agent algebra.

6 Normalizable Agent Algebra

As is customary in the study of algebraic systems, we can define expressions in terms of the operators that are defined in an agent algebra. In this section we define what it means for two agent expressions to be equivalent and prove that every expression can be transformed into an equivalent expression in a specific (normal) form.

Definition 6.1 (Agent Expressions). Let V be a set of variables, and let $\mathcal{Q}$ be an agent algebra. The set of agent expressions over $\mathcal{Q}$ is the least set $\mathcal{E}$ satisfying the following conditions:

Constant If $p \in \mathcal{Q}.D$, then $p \in \mathcal{E}$.

Variable If $v \in V$, then $v \in \mathcal{E}$.

Projection If $E \in \mathcal{E}$ and B is an alphabet, then $proj(B)(E) \in \mathcal{E}$.

Renaming If $E \in \mathcal{E}$ and r is a renaming function, then $rename(r)(E) \in \mathcal{E}$.

Parallel Composition If $E_1 \in \mathcal{E}$ and $E_2 \in \mathcal{E}$, then $E_1 \parallel E_2 \in \mathcal{E}$.

We denote by $sub(E)$ the set of all subexpressions of E, including E.

Agent expressions have no binding constructs (e.g. quantifiers). Therefore every variable in an agent expression is free. The set of free variables of an agent expressions can be defined by induction on the structure of expressions as follows.

Definition 6.2 (Free variables). Let E be an agent expression over $\mathcal{Q}$. The set $FV(E)$ of free variables of E is

- If $E = p$ for some $p \in \mathcal{Q}.D$, then $FV(E) = \emptyset$.
- If $E = v$ for some $v \in V$ then $FV(E) = \{v\}$.
- If $E = proj(B)(E_1)$ for some agent expression E_1 then $FV(E) = FV(E_1)$.
- If $E = rename(r)(E_1)$ for some agent expression E_1 then $FV(E) = FV(E_1)$.
- If $E = E_1 \parallel E_2$ for some agent expressions E_1 and E_2 then $FV(E) = FV(E_1) \cup FV(E_2)$.

We call an expression that has no free variables a *closed expression.*

Intuitively, an agent expression represents a particular agent in the underlying agent algebra once the variables have been given a value. Hence, to define the semantics of agent expressions we must first describe an assignment to the variables.

Definition 6.3 (Assignment). Let $\mathcal{Q}$ be an agent algebra and let V be a set of variables. An assignment of V on $\mathcal{Q}$ is a function $\sigma : V \mapsto \mathcal{Q}.D$.

The denotation $[\![E]\!]$ of an expression E is a function that takes an assignment σ and produces a particular agent in the agent algebra. Note however that since the operators in the agent algebra are partial functions, the denotation of an expression is also a partial function. The semantic function, the one that to each expression E associates the denotation $[\![E]\!]$ is, of course, a total function.

Definition 6.4 (Expression Evaluation). Let Σ be the set of all assignments. The denotation of agent expressions is given by the function $[\![-]\!] : \mathcal{E} \mapsto \Sigma \rightarrow \mathcal{Q}.D$ defined for each assignment $\sigma \in \Sigma$ by the following semantic equations:

- If $E = p$ for some $p \in \mathcal{Q}.D$, then $[\![E]\!]\sigma = p$.
- If $E = v$ for some $v \in V$ then $[\![E]\!]\sigma = \sigma(v)$.
- If $E = proj(B)(E_1)$ for some expression E_1 then $[\![E]\!]\sigma = proj(B)([\![E_1]\!]\sigma)$ if $[\![E_1]\!]\sigma$ is defined and $proj(B)([\![E_1]\!]\sigma)$ is defined. Otherwise $[\![E]\!]\sigma$ is undefined.
- If $E = rename(r)(E_1)$ for some expression E_1 then $[\![E]\!]\sigma = rename(r)([\![E_1]\!]\sigma)$ if $[\![E_1]\!]\sigma$ is defined and $rename(r)([\![E_1]\!]\sigma)$ is defined. Otherwise $[\![E]\!]\sigma$ is undefined.
- If $E = E_1 \parallel E_2$ for some expressions E_1 and E_2 then $[\![E]\!]\sigma = [\![E_1]\!]\sigma \parallel [\![E_2]\!]\sigma$ if both $[\![E_1]\!]\sigma$ and $[\![E_2]\!]\sigma$ are defined and $[\![E_1]\!]\sigma \parallel [\![E_2]\!]\sigma$ is defined. Otherwise $[\![E]\!]\sigma$ is undefined.

The following equivalent definition of expression evaluation highlights the fact that the semantic equations are syntax directed.

$$
\begin{aligned}
[\![p]\!]\sigma &= p \\
[\![v]\!]\sigma &= \sigma(v) \\
[\![\mathit{proj}(B)(E)]\!]\sigma &= \begin{cases} \mathit{proj}(B)([\![E]\!]\sigma) & \text{if } [\![E]\!]\sigma\downarrow \text{ and } \mathit{proj}(B)([\![E]\!]\sigma)\downarrow \\ \uparrow & \text{otherwise} \end{cases} \\
[\![\mathit{rename}(r)(E)]\!]\sigma &= \begin{cases} \mathit{rename}(r)([\![E]\!]\sigma) & \text{if } [\![E]\!]\sigma\downarrow \text{ and } \mathit{rename}(r)([\![E]\!]\sigma)\downarrow \\ \uparrow & \text{otherwise} \end{cases} \\
[\![E_1 \parallel E_2]\!]\sigma &= \begin{cases} [\![E_1]\!]\sigma \parallel [\![E_2]\!]\sigma & \text{if } [\![E_1]\!]\sigma\downarrow, [\![E_2]\!]\sigma\downarrow \text{ and } [\![E_1]\!]\sigma \parallel [\![E_2]\!]\sigma\downarrow \\ \uparrow & \text{otherwise} \end{cases}
\end{aligned}
$$

Since the semantic equations are syntax directed, the solution exists and is unique [12]. We extend the semantic function to sets of expressions and sets of assignments as follows.

Definition 6.5. Let $\mathcal{E}$ be a set of expressions and Σ' a set of assignments. We denote the possible evaluations of an expression in $\mathcal{E}$ under an assignment in Σ' as

$$[\![\mathcal{E}]\!]\Sigma' = \{[\![E]\!]\sigma : E \in \mathcal{E} \text{ and } \sigma \in \Sigma'\}.$$

Clearly, the value of an agent expression depends only on the value assigned by the assignment σ to the free variables.

Lemma 6.6 (Coincidence Lemma). Let E be an expression, and let σ_1 and σ_2 be two assignments such that for all $v \in FV(E)$, $\sigma_1(v) = \sigma_2(v)$. Then

$$[\![E]\!]\sigma_1 = [\![E]\!]\sigma_2.$$

Since an agent expression involves only a finite number of free variables, we use the notation $E[v_1, \ldots, v_n]$ to denote that E has free variables $v_1, \ldots, v_n$. In that case, we use the notation $E[p_1, \ldots, p_n]$ for $[\![E]\!]\sigma$ where $\sigma(v_i) = p_i$ for $1 \leq i \leq n$. Note also that if an agent expression has no free variables its value does not depend on the assignment σ.

When an expression contains variables it is possible to substitute another expression for the variable. [1]

Definition 6.7 (Expression Substitution). Let $\mathcal{Q}$ be an agent algebra and let E and E' be a agent expressions. The agent expression $E'' = E[v/E']$ obtained by substituting E' for v in E is defined by induction on the structure of expressions as follows:

- If $E = p$ for some $p \in \mathcal{Q}.D$, then $E'' = p$.
- If $E = w$ for some $w \in V$, $w \neq v$ then $E'' = w$.
- If $E = v$ then $E'' = E'$.
- If $E = \mathit{proj}(B)(E_1)$ for some expression E_1 then $E'' = \mathit{proj}(B)(E_1[v/E'])$.
- If $E = \mathit{rename}(r)(E_1)$ for some expression E_1 then $E'' = \mathit{rename}(r)(E_1[v/E'])$.

[1] While it is possible to define the simultaneous substitution of several expression for several variables, we limit the exposition to the single variable case to keep the notation simpler.

- If $E = E_1 \parallel E_2$ for some expressions E_1 and E_2 then $E'' = E_1[v/E'] \parallel E_2[v/E']$.

Expression substitution differs from expression evaluation in that substitution is a syntactic operation that returns a new expression, while evaluation is a semantic operation that returns a value. The two are related by the following result.

Lemma 6.8 (Substitution Lemma). Let $E_1[v]$ and $E_2[v]$ be two expressions in the variable v. Then for all agents p

$$E_1[v/E_2[v]][p] = E_1[E_2[p]]$$

We say that two expressions are equivalent if they have the same value for all possible assignments.

Definition 6.9 (Expression equivalence). Two expressions E_1 and E_2 are equivalent, written $E_1 \equiv E_2$, if and only if for all assignments σ, $[\![E_1]\!]\sigma = [\![E_2]\!]\sigma$.

In particular the above definition implies that if two expressions are equivalent then they are defined or undefined for exactly the same assignments. Notice also that because equivalence depends on the evaluation of the expression, two expressions may be equivalent relative to one agent algebra and not equivalent relative to another agent algebra. In other words, expression equivalence depends on the particular choice of underlying agent algebra.

Sometimes it is convenient to consider only a subset of the possible assignments. In that case we talk about equivalence modulo a set of assignments Σ'.

Definition 6.10 (Expression equivalence modulo Σ'). Let Σ' be a set of assignments. Two expressions E_1 and E_2 are equivalent modulo Σ', written $E_1 \equiv_{\Sigma'} E_2$, if and only if for all assignments $\sigma \in \Sigma'$, $[\![E_1]\!]\sigma = [\![E_2]\!]\sigma$.

We state the following results for expression equivalence only, but they extend to expression equivalence modulo Σ' in a straightforward way.

Lemma 6.11. Expression equivalence is an equivalence relation.

Because the semantics of expressions is syntax directed, the value of an expression depends only on the value of its subexpressions. Hence

Theorem 6.12. Expression equivalence is a congruence with respect to the operators of the agent algebra.

Proof: We show that if E_1 and E_2 are two agent expressions such that $E_1 \equiv E_2$, then for all alphabets B, $\mathit{proj}(B)(E_1) \equiv \mathit{proj}(B)(E_2)$. The cases for *rename* and $\parallel$ are similar. The proof consists of the following series of implications:

$E_1 \equiv E_2$

by definition 6.9

$\Leftrightarrow$ for all assignments σ, $[\![E_1]\!]\sigma = [\![E_2]\!]\sigma$

$\Rightarrow$ for all assignments σ, $\mathit{proj}(B)([\![E_1]\!]\sigma) = \mathit{proj}(B)([\![E_2]\!]\sigma)$

by definition 6.4

$\Leftrightarrow$ for all assignments σ, $[\![\mathit{proj}(B)(E_1)]\!]\sigma = [\![\mathit{proj}(B)(E_2)]\!]\sigma$

by definition 6.9

$\Leftrightarrow$ $\mathit{proj}(B)(E_1) \equiv \mathit{proj}(B)(E_2)$

$\square$

Lemma 6.13. Let E be an agent expression and let $\hat{E}$ be a subexpression of E. If $\hat{E} \equiv \hat{E}'$ for some $\hat{E}'$, then $E \equiv E'$, where E' is obtained from E by replacing $\hat{E}$ with $\hat{E}'$.

Proof: The proof is by induction on the structure of E. □

Equivalence is useful when we need to transform an expression into a form that is convenient for certain applications. In that case, we want to make sure that the transformations do not change the meaning (the semantics) of the expression. In this work we are particularly interested in a form where rename operator is applied first, then followed by the parallel composition operator, and finally by the projection operator. We call this the RCP normal form.

Definition 6.14 (RCP normal form). Let $\mathcal{Q}$ be an agent algebra and let $\mathcal{E}_0$ be the set of expressions:

$$\mathcal{E}_0 = \{p : p \in \mathcal{Q}.D\} \cup \{v : v \in V\}.$$

An agent expression E is said to be in RCP (i.e., rename, compose, project) normal form if it is of the form

$$E = \mathit{proj}(A)(\mathit{rename}(r_1)(E_1) \parallel \cdots \parallel \mathit{rename}(r_n)(E_n))$$

where A is an alphabet, $r_1, \ldots, r_n$ are renaming functions and $E_1, \ldots, E_n$ are expressions in $\mathcal{E}_0$.

The RCP normal form is similar to the normal form Dill defined for circuit algebra expressions [8]. In our case, however, we have extended the definition to expressions involving variables. This normal form corresponds to flattening the hierarchy: all agents are first instantiated using the rename operator, and are subsequently composed in parallel to form the entire system. The final projection is used to hide the internal signals.

A variation on the above example includes an additional projection operation before the agents are composed together. We restrict this normal form to expressions that do not involve variables, since we need to account for the alphabet of each of the agents.

Definition 6.15 (PRCP normal form). Let $p_1, \ldots, p_n$, with alphabets $A_1, \ldots, A_n$, respectively, be agents in some agent algebra. An expression involving the operations of the agent algebra is said to be in PRCP (i.e., project, rename, compose, project) normal form if it is of the form

$$\mathit{proj}(B)(\mathit{rename}(r_1)(\mathit{proj}(B_1)(p_1)) \parallel \cdots \parallel \mathit{rename}(r_n)(\mathit{proj}(B_n)(p_n))),$$

where for all j,

$$B_j = A_j \cap (B \cup A_1 \cup \cdots \cup A_{j-1} \cup A_{j+1} \cup \cdots \cup A_n).$$

In PRCP normal form, any local signal is projected away before an agent is composed with the rest of the system, where a local signal is one that is not in the alphabet of the expression (i.e., B) and is not in the alphabet of any of the other agents in the system.

In the rest of this section we will concentrate on the RCP normal form, since we will need to solve inequalities for variables in the application shown in section 9. In particular, we are interested in sufficient conditions that an algebra must satisfy in order for all expressions to have an equivalent RCP normal form. We will approach this problem in steps of increasing complexity. First we will consider expressions that do not involve variables, i.e. closed expressions. In that case, the expression is either defined or undefined, a condition that greatly simplifies the search for the normal form. As a second step, we will consider expressions where variables can only be assigned agents with a specific alphabet and that always make the expression defined or not defined. This is a case that is interesting in practice, and that does not require the stronger conditions of the general result. Finally we will explore a set of restrictions that are needed to obtain an equivalent normal form in the general case. We will see that alphabets again play a major role, and that they must be restricted in order for the appropriate renaming functions and projection operators to exist. All of this is formalized in the following definitions and results.

Definition 6.16 (Closed-Normalizable Agent Algebra). Let $\mathcal{Q}$ be an agent algebra. We say that $\mathcal{Q}$ is a *closed-normalizable* agent algebra if the renaming, projection parallel composition operators satisfy the axioms given below, where p and p' are elements of D and $A = \alpha(p)$ and $A' = \alpha(p')$.

A8. If $\mathit{rename}(r)(p)$ is defined, then it is equal to $\mathit{rename}(r\mid_{A\to r(A)})(p)$.

A9. $\mathit{rename}(r')(\mathit{rename}(r)(p)) = \mathit{rename}(r' \circ r)(p)$, if the left hand side of the equation is defined.

A10. If $\mathit{proj}(B)(p)$ is defined, then it is equal to $\mathit{proj}(B \cap A)(p)$.

A11. $\mathit{proj}(B)(\mathit{proj}(B')(p)) = \mathit{proj}(B \cap B')(p)$, if the left hand side of the equation is defined.

A12. If $\mathit{rename}(r)(\mathit{proj}(B)(p))$ is defined, then there exists a function r' such that

$$\mathit{rename}(r)(\mathit{proj}(B)(p)) = \mathit{proj}(r'(B))(\mathit{rename}(r')(p)).$$

A13. If $\mathit{proj}(B)(p)$ is defined, then there exists a function r such that $r(A) \cap A' \subseteq B$ and

$$\mathit{proj}(B)(p) = \mathit{proj}(B)(\mathit{rename}(r)(p)).$$

A14. $\mathit{rename}(r)(p \parallel p') = \mathit{rename}(r\mid_{A\to r(A)})(p) \parallel \mathit{rename}(r\mid_{A'\to r(A')})(p')$, if the left hand side of the equation is defined.

A15. $\mathit{proj}(B)(p \parallel p') = \mathit{proj}(B \cap A)(p) \parallel \mathit{proj}(B \cap A')(p')$, if $(A \cap A') \subseteq B$.

The axioms can be used to algebraically transform an expression into an equivalent RCP normal form, as the next result shows. Technically, since we are considering only sufficient and not necessary conditions, the term *normalizable* should apply to all agent algebras whose expression can be put in RCP normal form, whether or not they satisfy the axioms. In practice, we restrict our attention to only algebras that do satisfy the axioms for the purpose of normalization, and we therefore use the term to distinguish them from those that do not. We will continue to use this convention for the rest of this document, including the more general cases of normalizable agent algebras.

Theorem 6.17 (Normal Form - Closed Expressions). Let $\mathcal{Q}$ be a closed-normalizable agent algebra, and let E be a closed expression over $\mathcal{Q}$. Then E is equivalent to an expression in RCP normal form.

Proof: Let E be a closed expression. If E is undefined, then E is equivalent to any undefined closed expression in RCP normal form. If E is defined, then we construct an equivalent closed expression in RCP normal form by induction on the structure of expressions.

- Assume $E = p$ for some agent $p \in \mathcal{Q}.D$. Then $E = \mathit{proj}(A)(\mathit{rename}(id_A)(p))$ by A4 and A2.

- Assume $E = \textit{proj}(B)(E_1)$. Then, by induction, E_1 is equivalent to an expression $E_1' = \textit{proj}(B')(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n))$ in RCP normal form. Then

$$\begin{aligned} E &= \textit{proj}(B)(\textit{proj}(B')(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n))) \\ &\quad \text{By A11} \\ &= \textit{proj}(B \cap B')(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n)) \end{aligned}$$

which is in RCP normal form.

- Assume $E = \textit{rename}(r)(E_1)$. Then, by induction, E_1 is equivalent to $E_1' = \textit{proj}(B)(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n))$ in RCP normal form. Then

$$\begin{aligned} E &= \textit{rename}(r)(\textit{proj}(B)(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n))) \\ &\quad \text{By A12 there exists a renaming function } r' \text{ such that} \\ &= \textit{proj}(r'(B))(\textit{rename}(r')(\textit{rename}(r_1)(p_1) \parallel \cdots \parallel \textit{rename}(r_n)(p_n))) \\ &\quad \text{By A6, A14 and A3} \\ &= \textit{proj}(r'(B))(\textit{rename}(r' |_{r_1(A_1) \to r'(r_1(A_1))})(\textit{rename}(r_1)(p_1)) \parallel \cdots \\ &\qquad \parallel \textit{rename}(r' |_{r_n(A_n) \to r'(r_n(A_n))})(\textit{rename}(r_n)(p_n))) \\ &\quad \text{By A9} \\ &= \textit{proj}(r'(B))(\textit{rename}(r' |_{r_1(A_1) \to r'(r_1(A_1))} \circ r_1)(p_1) \parallel \cdots \\ &\qquad \parallel \textit{rename}(r' |_{r_n(A_n) \to r'(r_n(A_n))} \circ r_n)(p_n)) \end{aligned}$$

which is in RCP normal form.

- Assume $E = E_1 \parallel E_2$. Then, by induction, E_1 is equivalent to an RCP normal form $E_1' = \textit{proj}(B_1)(\textit{rename}(r_{11})(p_{11}) \parallel \cdots \parallel \textit{rename}(r_{n1})(p_{n1}))$ and E_2 is equivalent to an RCP normal form $E_2' = \textit{proj}(B_2)(\textit{rename}(r_{12})(p_{12}) \parallel \cdots \parallel \textit{rename}(r_{n2})(p_{n2}))$. Let A_1 be the alphabet of expression E_1'' such that $E_1' = \textit{proj}(B_1)(E_1'')$. We can assume, without loss of generality, that $B_1 \subseteq A_1$ and $B_2 \subseteq A_2$. By A13 there exists a function r_1 such that $r_1(A_1) \cap A_2 \subseteq B_1$ and

$$\textit{proj}(B_1)(E_1'') = \textit{proj}(B_1)(\textit{rename}(r_1)(E_1'')).$$

Similarly, there exists a function r_2 such that $r_2(A_2) \cap (A_1 \cup r_1(A_1)) \subseteq B_2$ and

$$\textit{proj}(B_2)(E_2'') = \textit{proj}(B_2)(\textit{rename}(r_2)(E_2'')).$$

By A10

$$\begin{aligned} \textit{proj}(B_1)(E_1'') &= \textit{proj}(B_1 \cap r_1(A_1))(\textit{rename}(r_1)(E_1'')) \\ \textit{proj}(B_2)(E_2'') &= \textit{proj}(B_2 \cap r_2(A_2))(\textit{rename}(r_2)(E_2'')) \end{aligned}$$

Note that since $r_1(A_1) \cap A_2 \subseteq B_1$, and since $B_2 \subseteq A_2$, also $r_1(A_1) \cap B_2 \subseteq B_1$. Thus also $r_1(A_1) \cap B_2 \subseteq B_1 \cap r_1(A_1)$. Hence

$$(B_1 \cup B_2) \cap r_1(A_1) = (B_1 \cap r_1(A_1)) \cup (B_2 \cap r_1(A_1)) = B_1 \cap r_1(A_1).$$

Likewise, since $r_2(A_2) \cap (A_1 \cup r_1(A_1)) \subseteq B_2$, also $r_2(A_2) \cap A_1 \subseteq B_2$ and therefore $r_2(A_2) \cap B_1 \subseteq B_2$. Thus $r_2(A_2) \cap B_1 \subseteq B_2 \cap r_2(A_2)$. Hence

$$(B_1 \cup B_2) \cap r_2(A_2) = (B_1 \cap r_2(A_2)) \cup (B_2 \cap r_2(A_2)) = B_2 \cap r_2(A_2).$$

Thus we have

$$\begin{aligned} \mathit{proj}(B_1)(E_1'') &= \mathit{proj}((B_1 \cup B_2) \cap r_1(A_1))(\mathit{rename}(r_1)(E_1'')) \\ \mathit{proj}(B_2)(E_2'') &= \mathit{proj}((B_1 \cup B_2) \cap r_2(A_2))(\mathit{rename}(r_2)(E_2'')) \end{aligned}$$

Moreover, since $r_2(A_2) \cap (A_1 \cup r_1(A_1)) \subseteq B_2$, we also have $r_2(A_2) \cap r_1(A_1) \subseteq B_2$, so that $r_2(A_2) \cap r_1(A_1) \subseteq B_1 \cup B_2$. Hence by A15

$$\begin{aligned} &\mathit{proj}(B_1)(E_1'') \parallel \mathit{proj}(B_2)(E_2'') = \\ &\quad = \mathit{proj}((B_1 \cup B_2) \cap r_1(A_1))(\mathit{rename}(r_1)(E_1'')) \parallel \\ &\quad\ \mathit{proj}((B_1 \cup B_2) \cap r_2(A_2))(\mathit{rename}(r_2)(E_2'')) \\ &\quad = \mathit{proj}(B_1 \cup B_2)(\mathit{rename}(r_1)(E_1'') \parallel \mathit{rename}(r_2)(E_2'')) \end{aligned}$$

By A9 and A6

$$\begin{aligned} \mathit{rename}(r_1)(E_1'') &= \mathit{rename}(r_1 \circ r_{11})(p_{11}) \parallel \cdots \parallel \mathit{rename}(r_1 \circ r_{n1}(p_{n1}) \\ \mathit{rename}(r_2)(E_2'') &= \mathit{rename}(r_2 \circ r_{12})(p_{12}) \parallel \cdots \parallel \mathit{rename}(r_2 \circ r_{n2}(p_{n2}) \end{aligned}$$

which proves the result.

□

If we consider an expression that involves variables, the axioms of agent algebras and the axioms of definition 6.16 may not be sufficient to ensure the existence of an equivalent normal form. Consider, for example, the expression

$$E = v.$$

We must find a renaming function r and an alphabet B such that

$$E \equiv \mathit{proj}(B)(\mathit{rename}(r)(v)).$$

The axioms are insufficient for two reasons. In the first place, A4 and A2 ensure the existence of an appropriate renaming function r and alphabet B for each agent. However, the algebra must be such that *the same* renaming function r and alphabet B can be used to construct an equivalent expression for all agents (or, at least, for the subset of agents that are assigned to v). The same is true of all the axioms that for all agents dictate the existence of a certain renaming function or alphabet. To make the algebra normalizable, the order of the quantifiers of these axioms must be exchanged, thus strengthening the requirements.

Secondly, we have dealt with the problem of definedness in theorem 6.17 by deriving a different normal form, according to whether the original closed expression is defined or not. However, unlike a closed expression, an expression may be defined or not defined depending on the assignment to its variables. To ensure equivalence, we must find an expression in normal form that is defined and not defined for exactly the same assignments. In the particular case above, since the expression $E = v$ is defined for all possible assignments to v, we must find a renaming function r and an alphabet B such that $proj(B)(rename(r)(v))$ is also always defined. Consequently, we must strengthen the axioms in two ways: by first requiring that the equalities that occur in the axioms are valid whether or not the left hand side is defined; and by introducing additional assumptions on the definedness of the operators to ensure the existence of the normal form.

However, one case that requires minimal strengthening of the axioms, and that is of great practical interest, is when variables are always assigned agents with the same alphabet, and such that the expression is always defined or always not defined. This is, for instance, the case in [8].

Definition 6.18 (Alpha-Normalizable Agent Algebra). Let $\mathcal{Q}$ be a normalizable agent algebra. We say that $\mathcal{Q}$ is *alpha-normalizable* if the renaming, projection and parallel composition operators satisfy the following axioms:

A16. For all alphabets A there exists a renaming function r' such that for all agents p such that $\alpha(p) = A$, if $rename(r)(proj(B)(p))$ is defined, then

$$rename(r)(proj(B)(p)) = proj(r'(B))(rename(r')(p)).$$

A17. For all alphabets A there exists a renaming function r such that for all agents p such that $\alpha(p) = A$, if $proj(B)(p)$ is defined, then $r(A) \cap A' \subseteq B$ and

$$proj(B)(p) = proj(B)(rename(r)(p)).$$

Note that, as in definition 6.16, the axioms are stated directly in terms of the operators and the agents of the algebra. However, by theorem 6.12, they can be used with expressions whenever every evaluation (possibly restricted to a set of assignments Σ') of the expressions involved satisfies the requirements of the axiom. In that case, the equality must be replaced by equivalence (possibly modulo Σ'). This remark applies especially to the proofs of theorem 6.19 and theorem 6.25 below.

Theorem 6.19 (Normal Form - Same Alphabet). Let $\mathcal{Q}$ be an alpha-normalizable agent algebra. Let E be an expression over $\mathcal{Q}$ and let Σ' be a set of assignments such that for all $\sigma_1, \sigma_2 \in \Sigma'$, $[\![E]\!]\sigma_1\downarrow$ if and only if $[\![E]\!]\sigma_2\downarrow$ and for all variables v, $\alpha(\sigma_1(v)) = \alpha(\sigma_2(v))$. Then E is equivalent modulo Σ' to an expression E' in RCP normal form.

Proof: The proof is similar to the proof of theorem 6.17. In fact, the transformations in the induction are the same and equally valid for every assignment (subject to the restrictions set forth in the statement of the theorem) and therefore preserve the evaluation of the expression no matter what agents replaces the variables. $\square$

In general, an equivalent RCP normal form for an expression that involves unrestricted variables and quantities does not exist. To see why, consider the following expression E:

$$E = proj(B)(p) \parallel v.$$

To normalize this expression we must rename p so that the signals that are in its alphabet and that are not in B do not conflict with the signals in v. The alphabet of v however depends on its assigned value. Thus, if we assume that for each signal in the master alphabet $\mathcal{Q}.\mathcal{A}$ there is an agent that has that signal in its alphabet, then there exists no renaming function with the above property. One could avoid conflicts by renaming v by folding the master alphabet into a subset of itself (this can be done only if the master alphabet is infinite), thus making the extra signals available for p. However, in general this changes the meaning of the expression (since v now appears renamed without being guarded by a projection), thus making it difficult to obtain an equivalent expression.

One could of course require that projection and parallel composition always commute. That, however, would not only unduly restrict the kinds of models of computation that can be studied as agent algebras, but, more importantly, would be contrary to the intuitive interpretation of the operations. Therefore, in the absence of conditions specific to particular agent algebras, we must restrict the extent of the alphabets that are used in the expression and in the assignments to the variables.

In the rest of this section we present sufficient conditions for the existence of an equivalent RCP normal form for expressions involving variables. In particular, we are looking for restrictions on the alphabet of agents while still maintaining full generality. This can be achieved by restricting the use of the master alphabet to only a subset of the available signals, as long as the subset has the same cardinality as the whole and still leaves enough signals available for the operations of renaming. As a consequence, the equivalence will be modulo some set of assignments Σ' that satisfies the restrictions.

In what follows we will make use of the following lemmas and definitions.

Lemma 6.20. Let $\mathcal{Q}$ be an agent algebra. Let E_1 and E_2 be two expression over $\mathcal{Q}$ and Σ' a set of assignments such that $E_1 \equiv_{\Sigma'} E_2$. Then $\alpha([\![E_1]\!]\Sigma') = \alpha([\![E_2]\!]\Sigma')$.

Proof: Let $a \in \alpha([\![E_1]\!]\Sigma')$. Then there exists an assignment $\sigma \in \Sigma'$ such that $[\![E_1]\!]\sigma = p$ and $a \in \alpha(p)$. But since $E_1 \equiv_{\Sigma'} E_2$, then also $[\![E_2]\!]\sigma = p$. Therefore $a \in \alpha([\![E_2]\!]\Sigma')$. The reverse direction is similar. □

Definition 6.21 (Small subset). Let W be a set and let B be a subset of W. We say that B is a small subset of W, written $B \Subset W$, if:

- W is infinite.
- The cardinality of the complement $W - B$ is greater than or equal to the cardinality of B.

Lemma 6.22. Let X and Z be sets such that $X \Subset Z$. Then there exists a set Y such that $X \Subset Y \Subset Z$.

Proof: Let Y_1 and Y_2 be two subsets of $Z - X$ of the same size such that $Z - X = Y_1 \cup Y_2$, and let $Y = X \cup Y_1$. Since $Z - X$ is infinite, $|Y_1| = |Y_2| = |Z - X|$. Since $|Z - X| \geq |X|$, also $|Y_1| \geq |X|$. Therefore $X \Subset Y$.

Since $X \Subset Y$, $|X| \leq |Y_1|$. Therefore, since $Y = X \cup Y_1$, $|Y| = |Y_1|$. Therefore also $|Y_2| = |Y|$. Hence $Y \Subset Y \cup Y_2 = Z$. □

We now have the vocabulary to state and prove the main result of this section.

Definition 6.23 (Normalizable Agent Algebra). Let $\mathcal{Q}$ be an agent algebra. We say that $\mathcal{Q}$ is a *normalizable* agent algebra if the renaming, projection and parallel composition operators satisfy the following axioms:

A18. For all alphabets A there exists an alphabet B such that $A \subseteq B$ and for all agents p such that $\alpha(p) \subseteq A$

$$p = rename(id_B)(p).$$

A19. For all alphabets B and agents p and for all alphabets A' such that $\alpha(p) \cap A' = \emptyset$

$$proj(B)(p) = proj(B \cup A')(p).$$

A20. For all alphabets B and B', and for all agents p

$$proj(B)(proj(B')(p)) = proj(B \cap B')(p).$$

A21. For all renaming functions r_1 and r_2 and for all agents p

$$rename(r_1)(rename(r_2)(p)) = rename(r_1 \circ r_2)(p)$$

where for all signals a,

$$(r_1 \circ r_2)(a) = \begin{cases} r_1(r_2(a)) & \text{if } r_1(r_2(a))\downarrow \\ \uparrow & \text{otherwise.} \end{cases}$$

A22. For all renaming functions r and for all alphabets B there exist renaming functions r' and r'' such that for all agents p

$$rename(r)(proj(B)(p)) = proj(r'(B))(rename(r'')(p)).$$

A23. For all alphabets B, for all alphabets A and for all alphabets A' such that $|(\mathcal{Q}.\mathcal{A} - A') - B| \geq |A - B|$ there exists a renaming function r such that $r(A) \cap A' \subseteq B$ and for all agents p such that $\alpha(p) \subseteq A$

$$proj(B)(p) = proj(B)(rename(r)(p)).$$

A24. For all renaming functions r and for all agents p_1 and p_2

$$rename(r)(p_1 \parallel p_2) = rename(r)(p_1) \parallel rename(r)(p_2).$$

A25. For all alphabets B and for all agents p_1 and p_2 such that $\alpha(p_1) \cap \alpha(p_2) \subseteq B$

$$proj(B)(p_1 \parallel p_2) = proj(B)(p_1) \parallel proj(B)(p_2).$$

Lemma 6.24. Let $\mathcal{Q}$ be a normalizable agent algebra. Then the following two statements are equivalent.

1. $\mathcal{Q}$ satisfies A19, i.e. for all alphabets B and agents p and for all alphabets A' such that $\alpha(p) \cap A' = \emptyset$

$$proj(B)(p) = proj(B \cup A')(p).$$

2. for all alphabets B and agents p and for all alphabets A' such that $\alpha(p) \subseteq A'$

$$proj(B)(p) = proj(B \cap A')(p).$$

Proof: For the forward implication, assume item 1 is true. Let B be an alphabet, p an agent and A' an alphabet such that $\alpha(p) \subseteq A'$. Let $K = B \cap A'$ and $J' = B - A'$. Then

$$\begin{aligned}
&\alpha(p) \subseteq A' \\
&\quad \text{since } J' \cap A' = \emptyset \\
\Leftrightarrow\ &\alpha(p) \cap J' = \emptyset \\
&\quad \text{by item 1} \\
\Rightarrow\ &\mathit{proj}(K)(p) = \mathit{proj}(K \cup J')(p) \\
\Leftrightarrow\ &\mathit{proj}(B \cap A')(p) = \mathit{proj}((B \cap A') \cup (B - A'))(p) \\
\Leftrightarrow\ &\mathit{proj}(B \cap A')(p) = \mathit{proj}(B)(p)
\end{aligned}$$

For the reverse implication, assume item 2 is true. Let B be an alphabet, p an agent and A' an alphabet such that $\alpha(p) \cap A' = \emptyset$. Let $K = B \cup A'$ and $J' = \alpha(p) \cup B$. Then

$$\begin{aligned}
&\alpha(p) \subseteq J' \\
&\quad \text{by item 2} \\
\Rightarrow\ &\mathit{proj}(K)(p) = \mathit{proj}(K \cap J')(p) \\
\Leftrightarrow\ &\mathit{proj}(B \cup A')(p) = \mathit{proj}((B \cup A') \cap (\alpha(p) \cup B))(p) \\
\Leftrightarrow\ &\mathit{proj}(B \cup A')(p) = \mathit{proj}((B \cap \alpha(p)) \cup (B \cap B) \cup (A' \cap \alpha(p)) \cup (A' \cap B))(p) \\
&\quad \text{since } B \cap \alpha(p) \subseteq B,\ A' \cap \alpha(p) = \emptyset \text{ and } A' \cap B \subseteq B \\
\Leftrightarrow\ &\mathit{proj}(B \cup A')(p) = \mathit{proj}(B)(p)
\end{aligned}$$

□

The following theorem shows that in a normalizable agent algebra any expression can be turned into an equivalent expression in RCP normal form when enough signals are available. The notation is simpler if we assume that the expression does not contain constants. This assumption is without loss of generality, since the case when an expression contains constants can be obtained by representing the constants with unique variables and by considering only assignments that assign the corresponding constant to the variables.

Theorem 6.25 (Normal Form). Let $\mathcal{Q}$ be an agent algebra such that $\mathcal{Q}.\mathcal{A}$ is infinite, and let E be an expression over $\mathcal{Q}$ that does not involve constants. Let Σ' be a set of assignments and $W \subseteq \mathcal{Q}.\mathcal{A}$ be an alphabet such that $\alpha([\![\mathit{sub}(E)]\!]\Sigma') \Subset W$. Then E is equivalent modulo Σ' to an expression E' in RCP normal form such that $\alpha([\![\mathit{sub}(E')]\!]\Sigma') \subseteq W$. In addition, if a variable v appears k times in E, then it appears k times in E'.

Proof: The proof uses the following result.

Lemma 6.26. Let $E = \mathit{rename}(r_1)(v_1) \| \cdots \| \mathit{rename}(r_n)(v_n)$ be an expression such that $\alpha([\![E]\!]\Sigma') = A$. Then

$$\mathit{rename}(r)(E) \equiv_{\Sigma'} \mathit{rename}(r \circ r_1)(v_1) \parallel \cdots \parallel \mathit{rename}(r \circ r_n)(v_n)$$

and both $\alpha([\![\mathit{rename}(r)(E)]\!]\Sigma') = r(A)$ and for all i, $\alpha([\![\mathit{rename}(r \circ r_i)(v_i)]\!]\Sigma') \subseteq r(A)$.

Proof: By A6 and A24

$$rename(r)(E) \equiv_{\Sigma'} rename(r)(rename(r_1)(v_1)) \parallel \cdots \parallel rename(r)(rename(r_n)(v_n)),$$

and by A21

$$rename(r)(E) \equiv_{\Sigma'} rename(r \circ r_1)(v_1) \parallel \cdots \parallel rename(r \circ r_n)(v_n).$$

Since $\alpha([\![E]\!]\Sigma') = A$, and by A5, for all i, $\alpha([\![rename(r_i)(v_i)]\!]\Sigma') \subseteq A$. Therefore by A3, $\alpha([\![rename(r)(E)]\!]\Sigma') = r(A)$ and for all i, $\alpha([\![rename(r \circ r_i)(v_i)]\!]\Sigma') \subseteq r(A)$. □

The proof is by induction on the structure of expressions.

- Let $E = v$. Let $A = \alpha([\![E]\!]\Sigma')$. By A18 there exists an alphabet B such that $A \subseteq B$ and for all p such that $\alpha(p) \subseteq A$

$$p = rename(id_B)(p).$$

Let now $\sigma \in \Sigma'$ be an assignment. Then by definition 6.4

$$\begin{aligned}
[\![v]\!]\sigma &= \sigma(v) \\
&\quad \text{Since } \alpha(\sigma(v)) \subseteq A \\
&= rename(id_B)(\sigma(v)) \\
&\quad \text{by A2, since } \alpha(\sigma(v)) = \alpha(rename(id_B)(\sigma(v))) \\
&= proj(\alpha(\sigma(v)))(rename(id_B)(\sigma(v))) \\
&\quad \text{by A19, since } \alpha(\sigma(v)) \cap A \subseteq \alpha(\sigma(v)) \\
&= proj(A)(rename(id_B)(\sigma(v))) \\
&\quad \text{by definition 6.4} \\
&= [\![proj(A)(rename(id_B)(v))]\!]\sigma.
\end{aligned}$$

Thus by definition 6.10

$$v \equiv_{\Sigma'} proj(A)(rename(id_B)(v)) = E'$$

which is in RCP normal form.

By inspection

$$sub(E') = \{v, rename(id_B)(v), proj(A)(rename(id_B)(v))\}.$$

By hypothesis,

$$\alpha([\![v]\!]\Sigma') = A \subseteq W.$$

Since $v \equiv_{\Sigma'} rename(id_B)(v)$, by lemma 6.20

$$\alpha([\![rename(id_B)(v)]\!]\Sigma') = A \subseteq W.$$

Since $v \equiv_{\Sigma'} proj(A)(rename(id_B)(v))$, by lemma 6.20

$$\alpha([\![proj(A)(rename(id_B)(v))]\!]\Sigma') = A \subseteq W.$$

Therefore, $\alpha([\![sub(E')]\!]\Sigma') \subseteq W$.

By inspection, if a variable v appears k times in E, it appears k times in the normal form.

- Let $E = proj(B)(E_1)$.

  By hypothesis, $\alpha([\![sub(E)]\!]\Sigma') \Subset W$. Then, since $sub(E_1) \subseteq sub(E)$, also $\alpha([\![sub(E_1)]\!]\Sigma') \Subset W$. Then, by induction, E_1 is equivalent to an expression E_1' in RCP normal form

  $$E_1' = proj(B')(rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n))$$

  and $\alpha([\![sub(E_1')]\!]\Sigma') \subseteq W$. Then by theorem 6.12

  $$E \equiv_{\Sigma'} proj(B)(proj(B')(rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n))).$$

  By A20, E is equivalent modulo Σ' to an expression E'

  $$E \equiv_{\Sigma'} E' = proj(B \cap B')(rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n))$$

  which is in RCP normal form.

  Let

  $$E_1'' = rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n).$$

  Then

  $$sub(E') = \{E'\} \cup sub(E_1'').$$

  Since by hypothesis $\alpha([\![E]\!]\Sigma') \subseteq W$, and since $E \equiv_{\Sigma'} E'$, by lemma 6.20

  $$\alpha([\![E']\!]\Sigma') \subseteq W.$$

  Since $sub(E_1'') \subseteq sub(E_1')$, and since $\alpha([\![E_1']\!]\Sigma') \subseteq W$,

  $$\alpha([\![sub(E_1'')]\!]\Sigma') \subseteq W.$$

  Therefore, $\alpha([\![sub(E')]\!]\Sigma') \subseteq W$.

  In addition, if a variable v appears k times in E, it appears k times in E_1, and therefore, by induction, it appears k times in E_1' and in the final normal form.

- Let $E = rename(r)(E_1)$.

  By hypothesis, $\alpha([\![sub(E)]\!]\Sigma') \Subset W$. Then, since $sub(E_1) \subseteq sub(E)$, also $\alpha([\![sub(E_1)]\!]\Sigma') \Subset W$. Then by induction E_1 is equivalent to an expression E_1' in RCP normal form

  $$E_1' = proj(B')(rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n))$$

and $\alpha(\llbracket\, sub(E_1')\,\rrbracket\Sigma') \subseteq W$.

Let $E_1'' = rename(r_1)(v_1) \parallel \cdots \parallel rename(r_n)(v_n)$. Then, by theorem 6.12

$$E \equiv_{\Sigma'} rename(r)(proj(B')(E_1'')).$$

By A22 there exist renaming function r' and r'' such that

$$E \equiv_{\Sigma'} proj(r'(B'))(rename(r'')(E_1'')).$$

Let now $A = \alpha(\llbracket\, rename(r'')(E_1'')\,\rrbracket\Sigma')$ and $B = r'(B') \cap A$. Since $\alpha(\llbracket\, E\,\rrbracket\Sigma') \Subset W$, by A1 and lemma 6.20 also $B \Subset W$. By lemma 6.24

$$E \equiv_{\Sigma'} proj(B)(rename(r'')(E_1'')).$$

Let now $A' = \mathcal{Q}.\mathcal{A} - W$. Then $(\mathcal{Q}.\mathcal{A} - A') - B = W - B$. Note that r'' is a bijection, and for any assignment $\sigma \in \Sigma'$, if $\llbracket\, rename(r'')(E_1'')\,\rrbracket\sigma$ is defined then also $\llbracket\, E_1''\,\rrbracket\sigma$ is defined. Thus, since $\alpha(\llbracket\, E_1''\,\rrbracket\Sigma') \subseteq W$, $|A| = |\alpha(\llbracket\, rename(r'')(E_1'')\,\rrbracket\Sigma')| \leq |\alpha(\llbracket\, E_1''\,\rrbracket\Sigma')| \leq |W|$. Hence, since $B \subseteq W$ and $B \subseteq A$, $|W - B| \geq |A - B|$. Therefore, by A23 there exists a renaming function r''' such that $r'''(A) \cap A' \subseteq B$ and

$$E \equiv_{\Sigma'} proj(B)(rename(r''')(rename(r'')(E_1''))).$$

By A21

$$E \equiv_{\Sigma'} proj(B)(rename(r''' \circ r'')(E_1'')).$$

By lemma 6.26

$$E \equiv_{\Sigma'} E' = proj(r'(B))(rename(r''' \circ r'' \circ r_1)(v_1) \parallel \cdots \parallel rename(r''' \circ r'' \circ r_n)(v_n))$$

which is in RCP normal form.

By inspection

$$\begin{aligned} sub(E') &= \{v_1, \ldots, v_n\} \cup \\ &= \cup \{rename(r''' \circ r'' \circ r_1)(v_1), \ldots, rename(r''' \circ r'' \circ r_n)(v_n)\} \cup \\ &= \cup \{rename(r''' \circ r'' \circ r_1)(v_1) \parallel \cdots \parallel rename(r''' \circ r'' \circ r_n)(v_n)\} \cup \\ &= \cup \{E'\} \end{aligned}$$

Since for all i, $v_i \in sub(E_1')$, and since $\alpha(\llbracket\, sub(E_1')\,\rrbracket\Sigma') \subseteq W$,

$$\alpha(\llbracket\, \{v_1, \ldots, v_n\}\,\rrbracket\Sigma') \subseteq W.$$

Note that $r'''(A) \cap A' \subseteq B$ implies $r'''(A) \subseteq W$. Therefore by lemma 6.26

$$\alpha(\llbracket\, \{rename(r''' \circ r'' \circ r_1)(v_1), \ldots, rename(r''' \circ r'' \circ r_n)(v_n)\}\,\rrbracket\Sigma') \subseteq r'''(A) \subseteq W$$
$$\alpha(\llbracket\, rename(r''' \circ r'' \circ r_1)(v_1) \parallel \cdots \parallel rename(r''' \circ r'' \circ r_n)(v_n)\,\rrbracket\Sigma') \subseteq r'''(A) \subseteq W.$$

Since by hypothesis $\alpha(\llbracket E \rrbracket \Sigma') \subseteq W$, and since $E \equiv_{\Sigma'} E'$, by lemma 6.20

$$\alpha(\llbracket E' \rrbracket \Sigma') \subseteq W.$$

Therefore, $\alpha(\llbracket \mathit{sub}(E') \rrbracket \Sigma') \subseteq W$.

In addition, if a variable v appears k times in E, it appears k times in E_1, and therefore, by induction, it appears k times in E_1' and in the final normal form.

- Let $E = E_1 \parallel E_2$.

  Let $A = \alpha(\llbracket E \rrbracket \Sigma')$.

  Since by hypothesis $A \Subset W$, by lemma 6.22, there exists an alphabet X such that $A \Subset X$ and $X \Subset W$.

  Then, since $\mathit{sub}(E_1) \subseteq \mathit{sub}(E)$ and $\mathit{sub}(E_2) \subseteq \mathit{sub}(E)$, also $\alpha(\llbracket E_1 \rrbracket \Sigma') \Subset X$ and $\alpha(\llbracket E_2 \rrbracket \Sigma') \Subset X$. Then, by induction, E_1 and E_2 are equivalent to expressions E_1' and E_2' in RCP normal form

$$\begin{aligned}
E_1' &= \mathit{proj}(B_1')(\mathit{rename}(r_{1,1})(v_{1,1}) \parallel \cdots \parallel \mathit{rename}(r_{1,n})(v_{1,n})) \\
E_2' &= \mathit{proj}(B_2')(\mathit{rename}(r_{2,1})(v_{2,1}) \parallel \cdots \parallel \mathit{rename}(r_{2,m})(v_{2,m}))
\end{aligned}$$

and $\alpha(\llbracket \mathit{sub}(E_1') \rrbracket \Sigma') \subseteq X$ and $\alpha(\llbracket \mathit{sub}(E_2') \rrbracket \Sigma') \subseteq X$.

Let

$$\begin{aligned}
E_1'' &= \mathit{rename}(r_{1,1})(v_{1,1}) \parallel \cdots \parallel \mathit{rename}(r_{1,n})(v_{1,n}) \\
A_1'' &= \alpha(\llbracket E_1'' \rrbracket \Sigma') \\
B_1 &= B_1' \cap A_1'' \\
E_2'' &= \mathit{rename}(r_{2,1})(v_{2,1}) \parallel \cdots \parallel \mathit{rename}(r_{2,m})(v_{2,m}) \\
A_2'' &= \alpha(\llbracket E_1'' \rrbracket \Sigma') \\
B_2 &= B_2' \cap A_2''
\end{aligned}$$

Then by theorem 6.12 and lemma 6.24

$$\begin{aligned}
E_1' &\equiv_{\Sigma'} \mathit{proj}(B_1)(E_1'') \\
E_2' &\equiv_{\Sigma'} \mathit{proj}(B_2)(E_2'')
\end{aligned}$$

Since $X \Subset W$, by lemma 6.22 there exists an alphabet Y such that $X \Subset Y$ and $Y \Subset W$.

Let $A_1' = \mathcal{Q}.\mathcal{A} - (W - Y)$ and $A_2' = \mathcal{Q}.\mathcal{A} - (Y - X)$. Clearly since $X \Subset Y \Subset W$ and $A_1'' \subseteq X$, $|W - Y| \geq |Y| \geq |X| \geq |A_1''|$. Therefore, since $B_1 \subseteq Y$, $|(\mathcal{Q}.\mathcal{A} - A_1') - B_1| = |(W - Y) - B_1| = |W - Y| \geq |A_1'' - B_1|$.

Similarly $|Y - X| \geq |X| \geq |A_2''|$. Therefore, since $B_2 \subseteq X$, $|(\mathcal{Q}.\mathcal{A} - A_2') - B_2| = |(Y - X) - B_2| = |Y - X| \geq |A_2'' - B_2|$.

Therefore, by A23 there exist renaming functions r'_1 and r'_2 such that $r'_1(A''_1) \cap A'_1 \subseteq B_1$, $r'_2(A''_2) \cap A'_2 \subseteq B_2$ and

$$\begin{aligned} E'_1 &\equiv_{\Sigma'} \mathit{proj}(B_1)(\mathit{rename}(r'_1)(E''_1)) \\ E'_2 &\equiv_{\Sigma'} \mathit{proj}(B_2)(\mathit{rename}(r'_2)(E''_2)) \end{aligned}$$

By definition, $B_2 \subseteq X \subseteq A'_1$ and $\alpha([\![\mathit{rename}(r'_1)(E''_1)]\!]\Sigma') = r'_1(A''_1)$, therefore since $r'_1(A''_1) \cap A'_1 \subseteq B_1$, also $r'_1(\alpha([\![\mathit{rename}(r'_1)(E''_1)]\!]\Sigma')) \cap B_2 \subseteq B_1$. Similarly, $r'_2(\alpha([\![\mathit{rename}(r'_1)(E''_1)]\!]\Sigma')) \cap B_1 \subseteq B_2$. Therefore by A19, denoting $B = B_1 \cup B_2$

$$\begin{aligned} E'_1 &\equiv_{\Sigma'} \mathit{proj}(B)(\mathit{rename}(r'_1)(E''_1)) \\ E'_2 &\equiv_{\Sigma'} \mathit{proj}(B)(\mathit{rename}(r'_2)(E''_2)) \end{aligned}$$

By the previous definitions $\alpha([\![\mathit{rename}(r'_1)(E''_1)]\!]\Sigma') \subseteq B_1 \cup (W - Y)$, and $\alpha([\![\mathit{rename}(r'_2)(E''_2)]\!]\Sigma') \subseteq B_2 \cup (Y - X)$. In addition, since $(W - Y) \cap (Y - X) = \emptyset$, $(B_1 \cup (W - Y)) \cap (B_2 \cup (Y - X)) = B_1 \cap B_2 \subseteq B$. Hence $\alpha([\![\mathit{rename}(r'_1)(E''_1)]\!]\Sigma') \cap \alpha([\![\mathit{rename}(r'_2)(E''_2)]\!]\Sigma') \subseteq B$. Therefore, by A25

$$E \equiv_{\Sigma'} \mathit{proj}(B)(\mathit{rename}(r'_1)(E''_1) \parallel \mathit{rename}(r'_2)(E''_2)).$$

By lemma 6.26

$$\begin{aligned} \mathit{rename}(r'_1)(E''_1) &\equiv_{\Sigma'} \mathit{rename}(r'_1 \circ r_{1,1})(v_{1,1}) \parallel \cdots \parallel \mathit{rename}(r'_1 \circ r_{1,n})(v_{1,n}) \\ \mathit{rename}(r'_2)(E''_2) &\equiv_{\Sigma'} \mathit{rename}(r'_2 \circ r_{2,1})(v_{2,1}) \parallel \cdots \parallel \mathit{rename}(r'_2 \circ r_{2,m})(v_{2,m}). \end{aligned}$$

Therefore by theorem 6.12

$$\begin{aligned} E \equiv_{\Sigma'} E' = {} & \mathit{proj}(B)(\mathit{rename}(r'_1 \circ r_{1,1})(v_{1,1}) \parallel \cdots \parallel \mathit{rename}(r'_1 \circ r_{1,n})(v_{1,n}) \parallel \\ & \parallel \mathit{rename}(r'_2 \circ r_{2,1})(v_{2,1}) \parallel \cdots \parallel \mathit{rename}(r'_2 \circ r_{2,m})(v_{2,m})) \end{aligned}$$

which is in RCP normal form.

By inspection

$$\begin{aligned} \mathit{sub}(E') &= \{v_{1,1}, \ldots, v_{1,n}\} \cup \\ &= \cup \{v_{2,1}, \ldots, v_{2,m}\} \cup \\ &= \cup \{\mathit{rename}(r'_1 \circ r_{1,1})(v_{1,1}), \ldots, \mathit{rename}(r'_2 \circ r_{2,m})(v_{2,m})\} \cup \\ &= \cup \{\mathit{rename}(r'_1 \circ r_{1,1})(v_{1,1}) \parallel \cdots \parallel \mathit{rename}(r'_2 \circ r_{2,m})(v_{2,m})\} \cup \\ &= \cup \{E'\} \end{aligned}$$

Since for all i, $v_{1,i} \in \mathit{sub}(E'_1)$, and since $\alpha([\![\mathit{sub}(E'_1)]\!]\Sigma') \subseteq W$,

$$\alpha([\![\{v_{1,1}, \ldots, v_{1,n}\}]\!]\Sigma') \subseteq W.$$

Similarly

$$\alpha([\![\{v_{2,1}, \ldots, v_{2,m}\}]\!]\Sigma') \subseteq W.$$

Note that $r_1'(A_1'') \cap A_1' \subseteq B_1$ implies $r_1'(A_1'') \subseteq W$, since $Q.\mathcal{A} - W \subseteq A_1'$ and $B_1 \subseteq A_1'' \subseteq W$. Therefore by lemma 6.26

$$\alpha([\![\{rename(r_1' \circ r_{1,1})(v_{1,1}), \ldots, rename(r_1' \circ r_{1,n})(v_{1,n})\}]\!]\Sigma') \subseteq r_1'(A_1'') \subseteq W$$
$$\alpha([\![rename(r_1' \circ r_{1,1})(v_{1,1}) \parallel \cdots \parallel rename(r_1' \circ r_{1,n})(v_{1,n})]\!]\Sigma') \subseteq r_1'(A_1'') \subseteq W$$

Similarly

$$\alpha([\![\{rename(r_2' \circ r_{2,1})(v_{2,1}), \ldots, rename(r_2' \circ r_{2,m})(v_{2,m})\}]\!]\Sigma') \subseteq r_2'(A_2'') \subseteq W$$
$$\alpha([\![rename(r_2' \circ r_{2,1})(v_{2,1}) \parallel \cdots \parallel rename(r_2' \circ r_{2,m})(v_{2,m})]\!]\Sigma') \subseteq r_2'(A_2'') \subseteq W$$

Since by hypothesis $\alpha([\![E]\!]\Sigma') \subseteq W$, and since $E \equiv_{\Sigma'} E'$, by lemma 6.20

$$\alpha([\![E']\!]\Sigma') \subseteq W.$$

Therefore, $\alpha([\![sub(E')]\!]\Sigma') \subseteq W$.

In addition, assume a variable appears m times in E. Then it appears j times in E_1 and k times in E_2 such that $m = j + k$. By induction, it appears j times in E_1' and k times in E_2', and therefore it appears $j + k$ times in the final normal form.

□

The rest of this section is devoted to proving the validity of some of the axioms for a few examples.

Example 6.27 (Alphabet Algebra). The alphabet agent algebra Q described in example 5.7 is a normalizable agent algebra. Here we show that A23 is satisfied.

Lemma 6.28. Q satisfies A23.

Proof: Let B, A and A' be alphabets over Q such that $|(Q.\mathcal{A} - A') - B| \geq |A - B|$. Let $r' : (A - B) \mapsto (Q.\mathcal{A} - A') - B$ be any injection from $A - B$ to $Q.\mathcal{A} - A' - B$. The injection exists because of the assumption on the cardinality of the sets. Then define an injection $r : A \mapsto Q.\mathcal{A}$ as follows:

$$r(a) = \begin{cases} r'(a) & \text{if } a \in A - B \\ id_A(a) & \text{otherwise} \end{cases}$$

Then if p is an agent such that $\alpha(p) \subseteq A$, and restricting the codomain of r to $r(A)$,

$$\begin{aligned} proj(B)(p) &= B \cap \alpha(p) \\ &= B \cap r(\alpha(p)) \\ &= proj(B)(rename(r)(p)). \end{aligned}$$

Therefore A23 is satisfied. □

Example 6.29 (IO Agent Algebra). The IO agent algebra Q described in example 5.10 is normalizable. Here we show that A25 is satisfied.

Lemma 6.30. Q satisfies A25.

Proof: Let B be an alphabet and let p_1 and p_2 be two agents such that $\alpha(p_1) \cap \alpha(p_2) \subseteq B$. Assume $proj(B)(p_1 \parallel p_2)$ is defined. Then by definition $(I_1 \cup I_2) - (O_1 \cup O_2) \subseteq B$. We now show that $I_1 \subseteq B$. Let $i \in I_1$ be a signal. Then by definition $i \notin O_1$. Assume $i \notin O_2$. Then $i \in (I_1 \cup I_2) - (O_1 \cup O_2)$ and therefore $i \in B$. On the other hand, assume $i \in O_2$. Then $i \in \alpha(p_1) \cap \alpha(p_2)$. Therefore $i \in B$. Hence $I_1 \subseteq B$. Similarly, $I_2 \subseteq B$. Therefore $proj(B)(p_1) \parallel proj(B)(p_2)$ is defined. In addition

$$proj(B)(p_1 \parallel p_2) = ((I_1 \cup I_2) - (O_1 \cup O_2), (O_1 \cup O_2) \cap B)$$
$$proj(B)(p_1) \parallel proj(B)(p_2) = ((I_1 \cup I_2) - ((O_1 \cup O_2) \cap B), (O_1 \cup O_2) \cap B)$$

Clearly

$$(I_1 \cup I_2) - (O_1 \cup O_2) \subseteq (I_1 \cup I_2) - ((O_1 \cup O_2) \cap B).$$

Let now $i \in (I_1 \cup I_2) - ((O_1 \cup O_2) \cap B)$. Then either $i \in I_1$ or $i \in I_2$ (or both). If $i \in I_1$ then $i \notin O_1$ and $i \notin O_2 \cap B$. If $i \in O_2$, then $i \in \alpha(p_1) \cap \alpha(p_2)$ and therefore $i \in B$. But then $i \in O_2 \cap B$, a contradiction. Hence $i \notin O_2$. Then $i \in (I_1 \cup I_2) - (O_1 \cup O_2)$. Similarly if $i \in I_2$. Therefore

$$(I_1 \cup I_2) - ((O_1 \cup O_2) \cap B) = (I_1 \cup I_2) - (O_1 \cup O_2)$$

and

$$proj(B)(p_1 \parallel p_2) = proj(B)(p_1) \parallel proj(B)(p_2).$$

Assume now $proj(B)(p_1 \parallel p_2)$ is not defined. Then there exists $i \in (I_1 \cup I_2) - (O_1 \cup O_2)$ such that $i \notin B$. But then either $i \in I_1$ or $i \in I_2$, and therefore either $I_1 \not\subseteq B$ or $I_2 \not\subseteq B$. Hence either $proj(B)(p_1)$ or $proj(B)(p_2)$ (or both) is undefined. Therefore $proj(B)(p_1) \parallel proj(B)(p_2)$ is undefined. □

Example 6.31 (Dill's IO Agent Algebra). The Dill's style IO agent algebra $\mathcal{Q}$ described in example 5.13 is not normalizable. In fact, it does not satisfy A18. The IO agent algebra described in example 6.29 is a generalization of $\mathcal{Q}$ that is normalizable.

However, Dill's style IO agent algebra is closed-normalizable, as described in [8]. It is also alpha-normalizable when the variable is restricted to assuming always the same value. If the algebra is used in isolation, this is obviously too restrictive for alpha-normalization to be of interest. However this is a useful property when used together with an appropriate behavior model for which alpha-normalization is non-trivial.

7 Conformance

Let p and p' be two agents in an ordered agent algebra. Intuitively, if we interpret the order as refinement, if $p \preceq p'$ then p can be substituted for p' in every context in which p' occurs. If this is the case we say that p *conforms* to p'. In this section we make this notion of substitutability precise. In our formalization, conformance is parameterized by a set of agents G, called a conformance set, and we only require that for p to conform to p', p can be substituted for p' for all contexts that evaluate in G. Intuitively, the set G identifies the set of contexts that "make sense" for a certain agent. The remaining contexts, which are of no interest for substitutability, are therefore ignored.

Conformance can be made more general by explicitly considering only a subset of the possible contexts. We call this notion *relative conformance.* In this section we will study these generalizations, and show the conditions under which relative conformance corresponds to conformance. We are particularly interested in composition contexts, also called *environments*, which are limited to the parallel composition with a single agent. Composition contexts will be the basis for studying mirror functions in the next section.

The concept of the context of an agent in a system plays a central role in the definition of conformance. It can be formalized using agent expressions.

Definition 7.1 (Expression Context). An expression context $E[\beta]$ is an expression with one free variable.

An expression context may or may not be defined depending on the agent that replaces the free variable. However, the property of $\top$-monotonicity of the operators of an ordered agent algebra transfers to expression contexts, as well.

Theorem 7.2. Let $\mathcal{Q}$ be an ordered agent algebra and $p \in \mathcal{Q}.D$ and $p' \in \mathcal{Q}.D$ be two agents such that $p \preceq p'$. For all expression contexts $E[\beta]$, if $E[p']$ is defined then $E[p]$ is also defined and $E[p] \preceq E[p']$.

Proof: The proof is by induction on the structure of the expression context.

- If $E[\beta] = q$ or $E[\beta] = \beta$ then the result follows directly from the hypothesis.
- Let $E[\beta] = \mathit{rename}(r)(E'[\beta])$ and assume $E[p']$ is defined. Then also $E'[p']$ is defined. By induction hypothesis $E'[p]$ is defined and $E'[p] \preceq E'[p']$. Since $\mathit{rename}(r)(E'[p'])$ is defined and *rename* is $\top$-monotonic, then $\mathit{rename}(r)(E'[p])$ is defined and $\mathit{rename}(r)(E'[p]) \preceq \mathit{rename}(r)(E'[p'])$.
- Let $E[\beta] = \mathit{proj}(B)(E'[\beta])$ and assume $E[p']$ is defined. Then also $E'[p']$ is defined. By induction hypothesis $E'[p]$ is defined and $E'[p] \preceq E'[p']$. Since $\mathit{proj}(B)(E'[p'])$ is defined and *proj* is $\top$-monotonic, then $\mathit{proj}(B)(E'[p])$ is defined and $\mathit{proj}(B)(E'[p]) \preceq \mathit{proj}(B)(E'[p'])$.
- Let $E[\beta] = E_1[\beta] \parallel E_2[\beta]$ and assume $E[p']$ is defined. Then also $E_1[p']$ and $E_2[p']$ are defined. By induction hypothesis $E_1[p]$ is defined and $E_1[p] \preceq E_1[p']$. Similarly, $E_2[p]$ is defined and $E_2[p] \preceq E_2[p']$. Since $E_1[p'] \parallel E_2[p']$ is defined and $\parallel$ is $\top$-monotonic, then $E_1[p] \parallel E_2[p']$ is also defined and $E_1[p] \parallel E_2[p'] \preceq E_1[p'] \parallel E_2[p']$. Similarly we conclude $E_1[p] \parallel E_2[p] \preceq E_1[p] \parallel E_2[p']$ and therefore since $\preceq$ is transitive $E_1[p] \parallel E_2[p] \preceq E_1[p'] \parallel E_2[p']$.

□

An ordered agent algebra $\mathcal{Q}$ has a conformance order parameterized by a set of agents G when the order corresponds to substitutability in the following sense.

Definition 7.3 (Conformance Order). Let $\mathcal{Q}$ be an ordered agent algebra and let G be a set of agents of $\mathcal{Q}$. We say $\mathcal{Q}$ has a *G-conformance order* if and only if for all agents p and p', $p \preceq p'$ if and only if for all expression contexts E, if $E[p'] \in G$ then $E[p] \in G$.

The implication in this definition is strong in the sense that if $E[p'] \in G$, then $E[p]$ must be defined (and be a member of G).

Each set of agents G induces a particular order, whether or not the algebra has a G-conformance order.

Definition 7.4. Let $\mathcal{Q}$ be an agent algebra and let G be a set of agents of $\mathcal{Q}$. We define $\mathcal{Q}.\mathit{conf}(G)$ to be the agent algebra that is identical to $\mathcal{Q}$ except that it has a G-conformance order.

We denote the order of $\mathcal{Q}.conf(G)$ with the symbol $\preceq_G$ and we say that G *induces* the order $\preceq_G$. In the rest of this section we will study some of the properties of $\mathcal{Q}.conf(G)$. In particular we are interested in characterizing when $\mathcal{Q}.conf(G)$ is an ordered agent algebra (i.e. the operators of projection, renaming and parallel composition are $\top$-monotonic) and when $\mathcal{Q}$ has a G-conformance order,

Lemma 7.5. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a subset of $\mathcal{Q}.D$. Consider the agent algebras $\mathcal{Q}.conf(G)$ and $\mathcal{Q}.conf(D)$. Then $\mathcal{Q}.conf(G)$ is an ordered agent algebra if and only if for all agents p and p',

$$p \preceq_G p' \Rightarrow p \preceq_D p'.$$

Proof: To show that $\mathcal{Q}.conf(G)$ is an ordered agent algebra, we need only show that its renaming, projection and parallel composition operators are $\top$-monotonic relative to its agent ordering. We prove the projection case (the others are similar).

Let p and p' be two agents such that $p \preceq_G p'$. We must show that if $proj(B)(p')$ is defined, then $proj(B)(p)$ is defined, and $proj(B)(p) \preceq_G proj(B)(p')$.

Since $p \preceq_G p'$, by hypothesis, $p \preceq_D p'$. Therefore, by definition 7.3, for all expression contexts E, if $E[p'] \in D$ then $E[p] \in D$. That is, if $E[p']$ is defined, then $E[p]$ is defined. Hence, if $E = proj(B)(\beta)$, if $proj(B)(p')$ is defined then $proj(B)(p)$ is defined.

Let now $E[\beta]$ be an expression context. We want to show that if $E[proj(B)(p')] \in G$, then $E[proj(B)(p)] \in G$. By lemma 6.8

$$E[proj(B)(p')] = E[proj(B)(\beta')[p']] = E[\beta/proj(B)(\beta)][p'].$$

Let now $E' = E[\beta/proj(B)(\beta)]$. Then

$$\begin{aligned} & E[proj(B)(p')] \in G \\ & \quad \Rightarrow \quad E'[p'] \in G \\ & \quad \Rightarrow \quad E'[p] \in G \\ & \quad \Rightarrow \quad E[proj(B)(p)] \in G \end{aligned}$$

Therefore, by definition 7.3

$$proj(B)(p) \preceq_G proj(B)(p').$$

Hence, the projection operator is $\top$-monotonic relative to $\preceq_G$.

Conversely, assume the operators are $\top$-monotonic relative to $\preceq_G$ and let p and p' be two agents such that $p \preceq_G p'$. Then, by theorem 7.2, for all expression contexts E, if $E[p']$ is defined, then $E[p]$ is defined. Hence, if $E[p'] \in D$, then $E[p] \in D$. Therefore, by definition 7.3, $p \preceq_D p'$. □

Corollary 7.6. Let $\mathcal{Q}$ be an ordered agent algebra. Then $\mathcal{Q}.conf(D)$ is an ordered agent algebra.

Although G can be any arbitrary set of agents, G must be downward closed relative to $\mathcal{Q}.\preceq$ in order for $\mathcal{Q}$ to have a G-conformance order.

Theorem 7.7. Let $\mathcal{Q}$ be an agent algebra and let G be a set of agents. Then G is downward closed relative to $\preceq_G$.

Proof: Let $p' \in G$ and let $p \preceq_G p'$. Consider the expression context $E = \beta$. Then clearly $E[p'] \in G$. But then, by definition 7.3, since $p \preceq_G p'$, also $E[p] = p \in G$. Therefore G is downward closed. $\square$

Corollary 7.8. Let $\mathcal{Q}$ be an ordered agent algebra. If $\mathcal{Q}$ has a G-conformance order, then G is downward closed relative to $\mathcal{Q}.\preceq$.

In the following we will explore the relationships between the order of $\mathcal{Q}$ and the orders induced by various conformance sets.

Theorem 7.9. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents. Then

$$p \preceq q \Rightarrow p \preceq_G q.$$

Proof: Since $p \preceq q$ and the operators are $\top$-monotonic, then by theorem 7.2 for all expression contexts E, if $E[q]$ is defined then $E[p]$ is defined. In addition, $E[p] \preceq E[q]$. Assume now that $E[q] \in G$. Then, since G is downward closed, also $E[p] \in G$. Therefore $p \preceq_G q$. $\square$

Notice that if G is downward closed, then the forward implication in definition 7.3 follows from theorem 7.9. If $\mathcal{Q}$ has a G-conformance order then the order is weak enough to ensure that the reverse implication also holds.

Corollary 7.10. If $\mathcal{Q}$ has a G-conformance order, then $\mathcal{Q}$ and $\mathcal{Q}.conf(G)$ are identical agent algebras.

The set of all agents D plays a special role, since it is always downward closed, no matter what order the agent algebra may have, and the order it induces always makes the operators $\top$-monotonic. The following theorem shows that given an agent algebra $\mathcal{Q}$, the ordered agent algebra $\mathcal{Q}.conf(D)$ has the weakest order that makes the operators $\top$-monotonic.

Corollary 7.11. Let $\mathcal{Q}$ be an ordered agent algebra. Then

$$p \preceq q \Rightarrow p \preceq_D q.$$

Proof: The result follows from theorem 7.9, since D is always downward closed. $\square$

Since the discrete order (i.e. the order such that $p \preceq p'$ if and only if $p = p'$) also makes the operator $\top$-monotonic, any order of an ordered agent algebra is bounded by the discrete order and by $\preceq_D$.

The following results show that if $\mathcal{Q}$ has the weakest conformance order (i.e. $\mathcal{Q} = \mathcal{Q}.conf(D)$), then any downward closed set of agents characterizes the order.

Corollary 7.12. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents such that $\mathcal{Q} = \mathcal{Q}.conf(G)$. Then

$$p \preceq_G q \Rightarrow p \preceq_D q.$$

Proof: Since $\mathcal{Q}$ is an ordered agent algebra, and $\mathcal{Q} = \mathcal{Q}.conf(G)$, also $\mathcal{Q}.conf(G)$ is an ordered agent algebra. Then the result follows from corollary 7.11. $\square$

Corollary 7.13. Let $\mathcal{Q}$ be an ordered agent algebra such that $\mathcal{Q} = \mathcal{Q}.conf(D)$, and let G be a downward closed set of agents such that $\mathcal{Q}.conf(G)$ is an ordered agent algebra. Then $\mathcal{Q} = \mathcal{Q}.conf(G)$.

Proof: Let p and p' be two agents such that $p \preceq p'$. The proof is composed of the following series of implications.

$$
\begin{aligned}
& p \preceq p' \\
& \quad \text{by theorem 7.9} \\
& \Rightarrow \quad p \preceq_G p' \\
& \quad \text{by corollary 7.12} \\
& \Rightarrow \quad p \preceq_D p' \\
& \quad \text{since } \mathcal{Q} = \mathcal{Q}.conf(D) \\
& \Rightarrow \quad p \preceq p'
\end{aligned}
$$

□

These results show that, in general, an ordered agent algebra $\mathcal{Q}$ can be characterized by several conformance sets. The particular choice of G influences the complexity of verifying the conformance relation, as we will see in the next few sections when we introduce relative conformance and mirror functions.

Note also that $\mathcal{Q}.conf(G)$ is not necessarily an agent algebra, in the sense that the operators may not be $\top$-monotonic relative to the agent ordering, even if they are $\top$-monotonic relative to the original ordering (see lemma 7.5). This is in practice not a problem, since we typically start from an ordered agent algebra, and then characterize its order in terms of a conformance set. In that case, since $\mathcal{Q} = \mathcal{Q}.conf(G)$, also $\mathcal{Q}.conf(G)$ is an ordered agent algebra.

7.1 Relative Conformance

In definition 7.3, conformance is defined in terms of all expression contexts. More in general, we can define conformance relative to a set of contexts.

Definition 7.14 (Relative Conformance). Let $\mathcal{Q}$ be an agent algebra and let G be a set of agents of $\mathcal{Q}$. We say $\mathcal{Q}$ has a *G-conformance order relative to a set of contexts $\mathcal{E}'$* if and only if for all agents p and p', $p \preceq p'$ if and only if for all expression contexts $E \in \mathcal{E}'$, if $E[p'] \in G$ then $E[p] \in G$.

A particularly interesting subset of contexts is the set of environments that consist of a parallel composition with an arbitrary agent.

Definition 7.15 (Composition Conformance). Let $\mathcal{Q}$ be an agent algebra and let G be a set of agents of $\mathcal{Q}$. We say $\mathcal{Q}$ has a *G-conformance order relative to composition* if and only if for all agents p and p', $p \preceq p'$ if and only if for all agents q, if $p' \parallel q \in G$ then $p \parallel q \in G$.

As with conformance, we define $\mathcal{Q}.conf(G, \mathcal{E}')$ to be the agent algebra that is identical to $\mathcal{Q}$ except that it has a G-conformance order relative to $\mathcal{E}'$. We denote the order of $\mathcal{Q}.conf(G, \mathcal{E}')$ with the symbol $\preceq_G^{\mathcal{E}'}$. In particular, $\mathcal{Q}.conf(G, \parallel)$ and $\preceq_G^{\parallel}$ denote G-conformance relative to composition.

Unlike conformance, $\mathcal{Q}.conf(G, \mathcal{E}')$ is not necessarily an ordered agent algebra even if $p \preceq_G^{\mathcal{E}'} p' \Rightarrow p \preceq_D^{\mathcal{E}'} p'$, since the operators of the algebra may not be $\top$-monotonic (cfr. lemma 7.5). In addition, if $\mathcal{Q}$ has a G-conformance order relative to $\mathcal{E}'$, then G is not necessarily downward closed (cfr. corollary 7.8).

Conformance implies relative conformance in the following sense.

Lemma 7.16. Let $\mathcal{Q}$ be an agent algebra and let $\mathcal{E}'$ be a set of contexts. Then for all agents p and p'

$$p \preceq_G p' \Rightarrow p \preceq_G^{\mathcal{E}'} p'.$$

Proof: Definition 7.14 is verified since the condition is by hypothesis true of all contexts. □

In particular, if $p \preceq_G p'$, then $p \preceq_G^{\parallel} p'$.

Despite the above result, if $\mathcal{Q}$ is an ordered agent algebra and $\mathcal{E}'$ is a set of contexts, $\mathcal{Q} = \mathcal{Q}.conf(G)$ does not necessarily imply $\mathcal{Q} = \mathcal{Q}.conf(G, \mathcal{E}')$. This is because the reverse implication above does not hold. However, if $\mathcal{Q}$ has a G-conformance order relative to some set of contexts $\mathcal{E}'$ and G is downward closed, then it also has a G-conformance order.

Theorem 7.17. Let $\mathcal{Q}$ be an ordered agent algebra, $\mathcal{E}'$ be a set of contexts and let G be a downward closed set of agents. Assume for all agents p and p',

$$p \preceq_G^{\parallel} p' \Rightarrow p \preceq p'.$$

Then $\mathcal{Q} = \mathcal{Q}.conf(G, \mathcal{E}') = \mathcal{Q}.conf(G)$.

Proof: We must show that for all agents p and p', $p \preceq p'$ if and only if $p \preceq_G p'$ if and only if $p \preceq_G^{\parallel} p'$. The result follows from the following circle of implications:

$$\begin{aligned}
&p \preceq p' \\
&\quad \text{by theorem 7.9, since } G \text{ is downward closed} \\
&\Rightarrow \quad p \preceq_G p' \\
&\quad \text{by lemma 7.16} \\
&\Rightarrow \quad p \preceq_G^{\mathcal{E}'} p' \\
&\quad \text{by hypothesis} \\
&\Rightarrow \quad p \preceq p'.
\end{aligned}$$

□

Corollary 7.18. Let $\mathcal{Q}$ be an ordered agent algebra, $\mathcal{E}'$ be a set of contexts and G a downward closed set of agents such that $\mathcal{Q} = \mathcal{Q}.conf(G, \mathcal{E}')$. Then $\mathcal{Q} = \mathcal{Q}.conf(G)$.

Proof: The result follows from theorem 7.17, since by hypothesis $p \preceq_G^{\mathcal{E}'} p' \Rightarrow p \preceq_G p'$. □

In particular, if $\mathcal{Q}$ has a G-conformance order relative to composition and G is downward closed, then it has a G-conformance order. In the examples that follow we will try to show, when possible, that conformance relative to composition corresponds exactly to conformance. When that is the case, it may be possible to find efficient ways to check the conformance relation, as we shall see in section 8.

Example 7.19 (Alphabet Algebra). Consider the agent algebra $\mathcal{Q}$ described in example 5.7, with the order such that $p \preceq p'$ if and only if $p \subseteq p'$. This order is the weakest order that makes the operators $\top$-monotonic, hence $\mathcal{Q} = \mathcal{Q}.conf(D)$. However, D does not characterize the order in terms of conformance relative to composition. Instead, conformance relative to composition induces the order such that every agent refines any other agent.

Theorem 7.20. For all agents p and p', $p \preceq_D^{\parallel} p'$.

Proof: The result follows from the fact that the conformance set in this case is the set of all agents D, and $\|$ is always defined. Therefore the condition in definition 7.15 is always satisfied. $\square$

In order to characterize the order in terms of conformance relative to composition we must consider the set $G = 2^{\mathcal{A}} - \mathcal{A}$, i.e. the set of all subsets of $\mathcal{A}$ except $\mathcal{A}$ itself. Then

Theorem 7.21. Let p and p' be two agents. Then the following statements are equivalent:

1. $p \subseteq p'$.
2. $p \preceq_G p'$.
3. $p \preceq_G^{\|} p'$.

Proof: We already know that $1 \Rightarrow 2$ (by theorem 7.9, since G is downward closed) and that $2 \Rightarrow 3$ (by lemma 7.16). The remaining implication is proved below.

Lemma 7.22. ($3 \Rightarrow 1$): Let p and p' be agents such that for all agents q, if $p' \| q \in G$ then $p \| q \in G$. Then $p \subseteq p'$.

Proof: Let p and p' be agents such that for all agents q, if $p' \| q \in G$ then $p \| q \in G$. By the definition of G, for all agents q, if $p' \| q \neq \mathcal{A}$ (i.e. $p' \| q \in G$), then $p \| q \neq \mathcal{A}$. Assume now, by contradiction, that $a \in p$ and $a \notin p'$. Consider $q = \mathcal{A} - p$. Then

$$p' \| q = p' \cup q = p' \cup (\mathcal{A} - p).$$

Since $a \notin p'$ and $a \notin \mathcal{A} - p$ (because $a \in p$), then $a \notin p' \| q$. Thus $p' \| q \neq \mathcal{A}$. Thus, by hypothesis, also $p \| q \neq \mathcal{A}$. However

$$p \| q = p \cup q = p \cup (\mathcal{A} - p) = \mathcal{A},$$

a contradiction. Thus $p \subseteq p'$. $\square$

$\square$

This is the only G that characterizes the order in terms of conformance relative to composition. In fact it is easy to show that for all $a \in \mathcal{A}$, the set $\mathcal{A} - \{a\}$ must be in G. Then, to characterize the order, G must be downward closed. Thus $G = 2^{\mathcal{A}} - \mathcal{A}$.

Example 7.23 (IO Agent Algebra). Consider the IO agent algebra $\mathcal{Q}$ defined in example 3.5 with the order defined in example 5.10. We now characterize the order in terms of conformance. Let $G = \{(I, O) : I = \emptyset\}$ be the conformance set that contains all agents that have no inputs. Then

Theorem 7.24. Let p and p' be IO agents. Then the following three statements are equivalent:

1. $p \preceq p'$ (i.e. $I \subseteq I'$ and $O = O'$).
2. $p \preceq_G p'$.
3. $p \preceq_G^{\|} p'$.

Proof: First we show that G is downward closed, then that $p \preceq_G^{\|} p'$ implies $p \preceq p'$. The result then follows from theorem 7.17.

Lemma 7.25. G is downward closed with respect to $\preceq$.

Proof: Let $p' \in G$. Then p' is of the form $(\emptyset, O')$ for some alphabet O'. Let $p = (I, O)$ be an agent such that $p \preceq p'$. Then, by the definition of the order, $I \subseteq \emptyset$, and therefore $I = \emptyset$. Hence $p \in G$. Therefore G is downward closed. □

Lemma 7.26. ($3 \Rightarrow 1$): Let p and p' be IO agents such that for all agents q, if $p' \| q \in G$ then $p \| q \in G$. Then $I \subseteq I'$ and $O = O'$.

Proof: We prove the result in steps.

($O \subseteq O'$) Assume, by contradiction, that there exists $o \in O$ such that $o \notin O'$. Consider $q = (O', I' \cup \{o\})$. Then $p' \| q$ is defined because $O' \cap (I' \cup \{o\}) = \emptyset$ since by hypothesis $O' \cap I' = \emptyset$ and $o \notin O'$. In addition $p' \| q \in G$. But then by hypothesis $p \| q$ is defined and $p \| q \in G$. However $\{o\} \subseteq O \cap (I' \cup \{o\})$, hence $O \cap (I' \cup \{o\}) \neq \emptyset$, a contradiction.

($O' \subseteq O$) Assume, by contradiction, that there exists $o \in O'$ such that $o \notin O$. Consider $q = (O', I')$. By hypothesis $o \notin I'$. Clearly $p' \| q$ is defined and $p' \| q \in G$, so by hypothesis also $p \| q \in G$ is defined and $p \| q \in G$. However

$$p \| q = ((I \cup O') - (O \cup I'), O \cup I')$$

However, since $o \in (I \cup O')$ and $o \notin (O \cup I')$, $p \| q \notin G$, a contradiction.

($I \subseteq I'$) Assume, by contradiction, that there exists $i \in I$ such that $i \notin I'$. By hypothesis we also have $i \notin O$. Consider $q = (O', I')$. Clearly $p' \| q$ is defined and $p' \| q \in G$, so by hypothesis also $p \| q \in G$ is defined and $p \| q \in G$. However

$$p \| q = ((I \cup O') - (O \cup I'), O \cup I') \ .$$

However, since $i \in (I \cup O')$ and $i \notin (O \cup I')$, $p \| q \notin G$, a contradiction.

□

□

Let us now consider the set of agents $G = \mathcal{Q}.D$ that consists of all agents. Then an expression evaluates in G if and only if the expression is defined. The following two theorems show that D still characterizes the order in terms of conformance, but it does not characterize the order in terms of conformance relative to composition.

Theorem 7.27. Let p and p' be IO agents. Then $p \preceq p'$ if and only if $p \preceq_D p'$.

Proof: The forward implication follows from theorem 7.9 since D is downward closed.

For the reverse implication, let $p = (I, O)$ and $p' = (I', O')$ be IO agents such that $p \preceq_D p'$. Then for all expression contexts E, if $E[p']$ is defined, then $E[p]$ is defined.

$(I \subseteq I')$ Consider the context $E = \mathit{proj}(I')(\beta)$. Then $E[p'] = \mathit{proj}(I')(p')$ is defined since $I' \subseteq I'$. Then also $E[p] = \mathit{proj}(I')(p)$ must be defined. Therefore $I \subseteq I'$.

$(O \subseteq O')$ Assume by contradiction that there exists $o \in O$ such that $o \notin O'$. Consider the agent $q = (\emptyset, \{o\})$ and the context $E = \beta \parallel q$. Then, since $O' \cap \{o\} = \emptyset$, $E[p'] = p' \parallel q$ is defined. Therefore also $E[p] = p \parallel q$ must be defined. But then $O \cap \{o\} = \emptyset$, a contradiction. Hence $O \subseteq O'$.

$(O' \subseteq O)$ Assume by contradiction that there exists $o \in O'$ such that $o \notin O$. Consider the agent $q = (\{o\}, \emptyset)$ and the context $E = \mathit{proj}(I')(\beta \parallel q)$. Then, since $I' \cap O' = \emptyset$ and $o \in O'$, $p' \parallel q = ((I' \cup \{o\}) - O', O') = (I', O')$. Therefore, since $I' \subseteq I'$, $E[p'] = \mathit{proj}(I')(p' \parallel q)$ is defined. Therefore also $E[p] = \mathit{proj}(I')(p \parallel q)$ must be defined. However, since $I \cap O = \emptyset$ and $o \notin O$, $p \parallel q = ((I \cup \{o\}) - O, O) = (I \cup \{o\}, O)$. In addition, $I \cup \{o\} \not\subseteq I'$, since $o \in O'$ implies $o \notin I'$, since $I' \cap O' = \emptyset$. Hence $\mathit{proj}(I')(p \parallel q)$ is not defined, a contradiction. Therefore $O' \subseteq O$.

□

Theorem 7.28. Let $p = (I, O)$ and $p' = (I', O')$ be IO agents. Then $p \preceq^{\parallel}_{D} p'$ if and only if $O \subseteq O'$.

Proof: For the forward direction, assume $p \preceq^{\parallel}_{D} p'$. Then for all agents q, if $p' \parallel q$ is defined, then also $p \parallel q$ is defined. Assume by contradiction that there exists $o \in O$ such that $o \notin O'$. Consider the agent $q = (\emptyset, \{o\})$. Then, since $O' \cap \{o\} = \emptyset$, $p' \parallel q$ is defined. Therefore, by definition of conformance, also $p \parallel q$ must be defined. But then $O \cap \{o\} = \emptyset$, a contradiction. Hence $O \subseteq O'$.

For the reverse direction, assume $O \subseteq O'$, and let $q = (I_q, O_q)$ be such that $p' \parallel q$ is defined. Then $O' \cap O_q = \emptyset$. Since $O \subseteq O'$, also $O \cap O_q = \emptyset$. Therefore $p \parallel q$ is defined. □

As expected, the order induced by D relative to composition does not make the operators T-monotonic. The above results also confirm that $\preceq$ is the weakest order such that the operators are T-monotonic.

Example 7.29 (Dill's IO Agent Algebra). Consider Dill's IO agent algebra $\mathcal{Q}$ defined in example 3.6 and example 5.13. The algebra is an ordered agent algebra if and only if $p \preceq p'$ corresponds to $p = p'$. Hence the only possible order is also the weakest possible order. Therefore $\mathcal{Q} = \mathcal{Q}.\mathit{conf}(D)$.

It is difficult however to characterize the order with conformance relative to composition. The following theorems characterize the conformance orders relative to composition induced by several conformance sets, and show that they do not correspond to the algebra's order.

Theorem 7.30. Let $G = \{(I, O) : I = \emptyset\}$ and let $p = (I, O)$ and $p' = (I', O')$ be agents. Then $p \preceq^{\parallel}_{G} p'$ if and only if $I \subseteq I'$ and $O = O'$.

Proof: The proof is the same as lemma 7.26. □

Theorem 7.31. Let $G = D$ and let $p = (I, O)$ and $p' = (I', O')$ be agents. Then $p \preceq^{\parallel}_{D} p'$ if and only if $O \subseteq O'$.

Proof: The proof is the same as theorem 7.28. □

Theorem 7.32. Let $G = \{(\emptyset, \mathcal{Q}.\mathcal{A})\}$ and let $p = (I, O)$ and $p' = (I', O')$ be agents. Then $p \preceq_G^{\parallel} p'$ if and only if $O = O'$.

Proof: For the forward direction, assume $p \preceq_G^{\parallel} p'$. Consider the agent $q = (O', \mathcal{A} - O')$. Then $p' \parallel q = (\emptyset, \mathcal{A}) \in G$. Hence also $p \parallel q$ must be defined, and therefore $O \cap (\mathcal{A} - O') = \emptyset$. But then $O \subseteq O'$. In addition $p \parallel q \in G$, and therefore $O \cup (\mathcal{A} - O') = \mathcal{A}$. But then $O \supseteq O'$. Hence $O = O'$.

For the reverse direction, assume $O = O'$. Let $q = (I_q, O_q)$ be an agent. If $p' \parallel q$ is defined, then $O' \cap O_q = \emptyset$. But then also $O \cap O_q = \emptyset$, and therefore also $p \parallel q$ is defined. In addition, if $p' \parallel q \in G$ then it must be $O' \cap O_q = \emptyset$ (for the composition to be defined) and $O' \cup O_q = \mathcal{A}$, and therefore $O_q = \mathcal{A} - O'$. Hence also $p \parallel q \in G$. Therefore $p \preceq_G^{\parallel} p'$. □

Example 7.33 (Typed IO Agent Algebra). Consider the Typed IO agent algebra $\mathcal{Q}$ defined in example 3.7 with the order defined in example 5.15. We would now like to characterize the order in terms of a conformance set. This can be done if we choose G to be the set of agents p such that *inputs*$(p) = \emptyset$.

Theorem 7.34. Let p and p' be Typed IO agents. Then the following three statements are equivalent:

1. $p \preceq p'$.
2. $p \preceq_G p'$.
3. $p \preceq_G^{\parallel} p'$.

Proof: We already know that $1 \Rightarrow 2$ (by theorem 7.9, since G is downward closed) and that $2 \Rightarrow 3$ (by lemma 7.16). The remaining implication is proved below.

Lemma 7.35. ($3 \Rightarrow 1$): Let p and p' be agents such that for all agents q, if $p' \parallel q \in G$ then $p \parallel q \in G$. Then $p \preceq p'$.

Proof: It is easy to adapt the proof of lemma 7.26 to show that *inputs*$(p) \subseteq$ *inputs*(p') and that *outputs*$(p) =$ *outputs*(p'). To prove the rest of the theorem, let $q = f_q$ be the agent such that for all $a \in \mathcal{Q}.\mathcal{A}$

$$f_q(a) = \begin{cases} (c_O, v) & \text{if } f'(a) = (c_I, v) \\ (c_I, v) & \text{if } f'(a) = (c_O, v) \\ c_U & \text{otherwise} \end{cases}$$

so that *inputs*$(q) =$ *outputs*(p') and *outputs*$(q) =$ *inputs*(p'). Then clearly $p' \parallel q$ is defined, and by definition of $\parallel$, $p' \parallel q \in G$. Thus, by hypothesis, also $p \parallel q \in G$. Let now $a \in \mathcal{Q}.\mathcal{A}$. If $a \in$ *inputs*(p), then $a \in$ *inputs*(p') and $a \in$ *outputs*(q). Since $p \parallel q$ is defined, then $f_q(a).v \subseteq f(a).v$, and thus $f'(a).v \subseteq f(a).v$. Similarly, if $a \in$ *outputs*(p), then $a \in$ *outputs*(p') and $a \in$ *inputs*(q). Since $p \parallel q$ is defined, then $f(a).v \subseteq f_q(a).v$, and thus $f(a).v \subseteq f'(a).v$. Thus $p \preceq p'$. □

□

8 Mirrors

In this section we address the problem of checking in an ordered agent algebra whether two agents are related by the order. If the algebra has a G-conformance order, then the problem reduces to verifying the condition for conformance. This problem however is rather expensive, since it requires considering all possible contexts. When conformance corresponds to conformance relative to composition then we need only check contexts that consist of parallel compositions with other agents. We define an *environment* of an agent to be a composition context. In this section we show how, in certain cases, it is possible to construct for each agent a single environment that determines the order. We call this environment the *mirror* of an agent.

Definition 8.1 (Mirror Function). Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents of $\mathcal{Q}$. Then, *$\mathcal{Q}$ has a mirror function relative to G* if and only if

1. $\mathcal{Q}$.*mirror* (which we may simply write as "*mirror*" when there is no ambiguity about what agent algebra is being considered) is a partial function from D to D,
2. $mirror(p)$ is defined if and only if there exists q such that $p \parallel q \in G$,
3. $p \preceq p'$ if and only if either $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$.

When an ordered agent algebra $\mathcal{Q}$ has a mirror function relative to some set of agents G, then we can verify that $p \preceq p'$ by simply looking at the composition $p \parallel mirror(p')$. We assume that computing the mirror, the composition and verifying the membership in G is computationally less expensive than checking that $p \preceq p'$ directly.

In the rest of this section we will explore the consequences of having a mirror function. Later, we will explore necessary and sufficient conditions for an ordered agent algebra to have a mirror function.

Lemma 8.2. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. For all agents p, if $mirror(p)$ is defined, then $p \parallel mirror(p) \in G$.

Proof: Since $\preceq$ is reflexive, $p \preceq p$. By definition 8.1, this implies $mirror(p)$ is undefined or $p \parallel mirror(p) \in G$. □

Theorem 8.3. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. For all agents p, if $mirror(p)$ is defined, then $mirror^2(p)$ is also defined.

Proof: Assume $mirror(p)$ is defined. By lemma 8.2, $p \parallel mirror(p) \in G$. This implies that there exists a p' (namely p) such that $p' \parallel mirror(p) \in G$. By definition 8.1, this implies that $mirror^2(p)$ is defined. □

Corollary 8.4. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. For all agents p, if $mirror(p)$ is defined, then $mirror^n(p)$ is also defined, for any positive integer n.

Proof: By induction on n. □

Lemma 8.5. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. Let $p \parallel q \in G$ and let $p' \preceq q$. Then $p \parallel p' \in G$.

Proof: By hypothesis $p \parallel q$ is defined and $p' \preceq q$. Then, since parallel composition is $\top$-monotonic, also $p \parallel p'$ is defined and $p \parallel p' \preceq p \parallel q$. Therefore, since G is downward closed, $p \parallel p' \in G$. □

Lemma 8.6. Let Q be an ordered agent algebra with a mirror function relative to G. Let p and q be agents such that $mirror(p)$ and $mirror(q)$ are both defined. Then,

$$mirror(p) \preceq q \Leftrightarrow mirror(q) \preceq p.$$

Proof: The proof is composed of the following series of double implications:

$mirror(p) \preceq q$

by definition 8.1, since $mirror(q)$ is defined

$\Leftrightarrow \quad mirror(p) \parallel mirror(q) \in G$

since $\parallel$ is commutative by A7

$\Leftrightarrow \quad mirror(q) \parallel mirror(p) \in G$

by definition 8.1

$\Leftrightarrow \quad mirror(q) \preceq p$

□

The term mirror is justified by the following result.

Theorem 8.7. Let Q be an ordered agent algebra with a mirror function relative to G. Let p be an agent and assume $mirror(p)$ is defined. Then $mirror^2(p)$ is defined and

$$p \approx mirror^2(p).$$

Proof: It follows from corollary 8.4 that $mirror^2(p)$ is defined and $mirror^3(p)$ is defined. By definition 5.3, it is sufficient to show that $mirror^2(p) \preceq p$ and $p \preceq mirror^2(p)$.

Lemma 8.8. $mirror^2(p) \preceq p$.

Proof: $mirror(p) \preceq mirror(p)$, since $\preceq$ is reflexive. Thus, by lemma 8.6, $mirror^2(p) \preceq p$. □

Lemma 8.9. $p \preceq mirror^2(p)$.

Proof: $p \preceq p$, since $\preceq$ is reflexive. We complete the proof with the following chain of implications.

$p \preceq p$

by definition 8.1, since $mirror(p)$ is defined

$\Leftrightarrow \quad p \parallel mirror(p) \in G$

by lemma 8.8 and lemma 8.5

$\Rightarrow \quad p \parallel mirror^3(p) \in G$

by definition 8.1

$\Leftrightarrow \quad p \preceq mirror^2(p)$.

□

□

Theorem 8.10. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. Let p and q be agents such that $mirror(p)$ and $mirror(q)$ are defined. Then,

$$p \preceq q \Leftrightarrow mirror(q) \preceq mirror(p).$$

Proof: By corollary 8.4 we know that $mirror^2(q)$ is defined. By applying lemma 8.6 to q and $mirror(p)$, we get

$$mirror(q) \preceq mirror(p) \Leftrightarrow mirror^2(p) \preceq q.$$

By theorem 8.7, we know that $q \approx mirror^2(q)$. Thus

$$mirror^2(p) \preceq q \Leftrightarrow q \preceq p,$$

which implies the desired result. □

Since mirrors reduce the problem of verifying conformance to a single composition environment, it is not surprising that their existence is related to G-conformance relative to composition. In fact, the mirror of an agent has an exact characterization in terms of the G-conformance order relative to composition and the greatest element of a certain set of agents.

Let G be a conformance set and let p and q be agents. If $p \parallel q \in G$ then we say that q is compatible (or G-compatible if we want to emphasize the conformance set) with p. We call the set of agents that are compatible with p the *compatibility set* of p.

Definition 8.11 (Compatibility Set). Let $\mathcal{Q}$ be an ordered agent algebra and G a downward closed set of agents. The *G-compatibility set* of an agent p, written $cmp(p)$, is defined as follows:

$$cmp(p) = \{q : p \parallel q \in G\}$$

The compatibility set gets larger as the agents are more refined according to the order of the algebra, as shown in the next theorem.

Lemma 8.12. Let $\mathcal{Q}$ be an ordered agent algebra and G a downward closed set of agents. Let p and p' be agents such that $p \preceq p'$. Then

$$cmp(p') \subseteq cmp(p).$$

Proof: We show that if $q \in cmp(p')$, then $q \in cmp(p)$. The proof consists of the following series of implications.

$q \in cmp(p')$

by definition 8.11

$\Leftrightarrow \quad p' \parallel q \in G$

since $\parallel$ is $\top$-monotonic and $p \preceq p'$

$\Rightarrow \quad p \parallel q \preceq p' \parallel q$

since G is downward closed

$\Rightarrow \quad p \parallel q \in G$

by definition 8.11

$\Leftrightarrow \quad q \in cmp(p).$

□

When an agent algebra has a G-conformance order relative to composition, the order is determined by the compatibility set of each agent.

Lemma 8.13. Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Then for all agents p and p',

$$p \preceq p' \Leftrightarrow p \parallel cmp(p') \subseteq G,$$

where $\parallel$ has been naturally extended to sets.

Proof: The result follows directly from definition 7.15. □

Since the operators of an ordered agent algebra are $\top$-monotonic, the maximal elements of the compatibility set are sufficient to completely determine the order.

Lemma 8.14. Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Then for all agents p and p',

$$p \preceq p' \Leftrightarrow \text{for all } q \text{ such that } q \text{ is maximal in } cmp(p), p \parallel q \in G.$$

Proof: The forward implication is simply a special case of lemma 8.13.

For the reverse implication, let $q \in cmp(p')$ be an agent. Then there exists $q' \in cmp(p')$ such that q' is maximal and $q \preceq q'$. By hypothesis, $p \parallel q' \in G$. Note that G is downward closed, since $\mathcal{Q}$ has a G-conformance order relative to composition. Hence, since $\parallel$ is $\top$-monotonic and G is downward closed, also $p \parallel q \in G$. Therefore $p \parallel cmp(p') \subseteq G$. The desired result then follows from lemma 8.13. □

We often denote the set of maximal elements of $cmp(p)$ as $maxcmp(p)$.

Lemma 8.14 suggests that the mirror of an agent should be found among the maximal elements of the compatibility set. In fact, since the mirror alone is sufficient to determine the order, it suggests that the mirror should be the greatest element of the compatibility set. In the following we will make the relationship between the mirror and the compatibility set more precise.

Theorem 8.15. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. If $p \parallel q \in G$, then $q \preceq mirror(p)$.

Proof: The proof is composed of the following implications:

$p \parallel q \in G$

by definition 8.1

$\Leftrightarrow$ $mirror(p)$ is defined

by lemma 8.8

$\Rightarrow$ $mirror^2(p) \preceq p$

by lemma 8.5

$\Rightarrow$ $mirror^2(p) \parallel q \in G$

by definition 3.1 (commutativity)

$\Leftrightarrow$ $q \parallel mirror^2(p) \in G$

by definition 8.1

$\Leftrightarrow$ $q \preceq mirror(p)$

□

When an agent algebra has a mirror function relative to a set G, then it has a G-conformance order relative to composition and a G-conformance order, as shown by the next results.

Theorem 8.16. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents. If $\mathcal{Q}$ has a mirror function relative to G, then $\mathcal{Q}$ has a G-conformance order relative to composition.

Proof: We must show that for all agents p and p', $p \preceq p'$ if and only $p \preceq_G^{\|} p'$.

The forward implication follows from theorem 7.9 and lemma 7.16 since G is downward closed.

For the reverse implication we consider two cases. Assume $mirror(p')$ is not defined. Then, by definition 8.1, $p \preceq p'$.

Assume $mirror(p')$ is defined. Then, by lemma 8.2, $p' \parallel mirror(p') \in G$. Then, since $p \preceq_G^{\|} p'$, also $p \parallel mirror(p') \in G$. Therefore, by definition 8.1, $p \preceq p'$. □

Corollary 8.17. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents. If $\mathcal{Q}$ has a mirror function relative to G, $\mathcal{Q}$ has a G-conformance order.

Proof: The result follows from theorem 8.16 and theorem 7.17. □

In other words, when an algebra $\mathcal{Q}$ has a mirror function relative to G, both G-conformance and G-conformance relative to composition characterize the order.

We can now completely characterize the mirror function in terms of conformance and the compatibility sets.

Theorem 8.18 (Mirror Characterization). Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents. Then the following two statement are equivalent:

1. $\mathcal{Q}$ has a mirror function relative to G.
2. $\mathcal{Q}$ has a G-conformance order relative to composition, and for all agents p', $cmp(p')$ is either empty or if it is not empty it has a greatest element.

Proof: Assume $\mathcal{Q}$ has a mirror function relative to G. Then, by theorem 8.16, $\mathcal{Q}$ has a G-conformance order realtive to composition. In addition, let p' be an agent. If $mirror(p')$ is undefined, then, by definition 8.1, $cmp(p')$ is empty. Otherwise, if $mirror(p')$ is defined, then, by definition 8.1, $cmp(p')$ is not empty, and, by theorem 8.15, $mirror(p')$ is its greatest element.

Conversely, assume $\mathcal{Q}$ has a G-conformance order relative to composition, and for all agents p', $cmp(p')$ is either empty or if it is not empty it has a greatest element. We show that the function

$$mirror(p') = \begin{cases} \max(cmp(p')) & \text{if } cmp(p') \neq \emptyset \\ \uparrow & \text{if } cmp(p') = \emptyset \end{cases}$$

is a mirror function relative to G.

Clearly $mirror$ is a partial function, and $mirror(p')$ is defined if and only if there exists an agent q such that $p' \parallel q \in G$. It remains to be shown that for all agents p and p', $p \preceq p'$ if and only if $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$.

Assume $p \preceq p'$. If $mirror(p')$ is undefined we are done. Assume $mirror(p')$ is defined. Since $mirror(p') \in cmp(p')$, $p' \parallel mirror(p') \in G$. But $\mathcal{Q}$ has a G-conformance order relative to composition, hence $p \preceq p'$ if and only if for all q, if $p' \parallel q \in G$ then $p \parallel q \in G$. Therefore $p \parallel mirror(p') \in G$, since by hypothesis $p \preceq p'$.

Conversely assume $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$. If $mirror(p')$ is undefined, then $cmp(p') = \emptyset$, and therefore for all agents q, if $p' \parallel q \in G$ then $p \parallel q \in G$ vacuously. Hence $p \preceq_G^{\parallel} p'$, and since $\mathcal{Q}$ has a G-conformance order relative to composition, also $p \preceq p'$.

On the other hand, assume $mirror(p')$ is defined and $p \parallel mirror(p') \in G$. Let q be an agent such that $p' \parallel q \in G$. Then, by our definition of $mirror(p')$, $q \preceq mirror(p')$, since $mirror(p')$ is the greatest compatible agent. Then $p \parallel q \in G$, since $p \parallel mirror(p') \in G$, $q \preceq mirror(p')$, $\parallel$ is $\top$-monotonic and G is downward closed. Hence $p \preceq_G^{\parallel} p'$, and since $\mathcal{Q}$ has a G-conformance order relative to composition, also $p \preceq p'$. □

These results show that the mirror of an agent corresponds to the greatest element of the compatibility set. For general preordered agent algebra, the compatibility set may have several different greatest element. In that case there is some flexibility in the choice of the mirror function. However, if the algebra is partially ordered (i.e. the order is antisymmetric), the greatest element is unique. Hence, if a mirror function exists, it is uniquely determined.

Theorem 8.19. Let $\mathcal{Q}$ be a partially ordered agent algebra. If $\mathcal{Q}$ has a mirror function relative to G, then the mirror function is uniquely determined.

Proof: Assume $\mathcal{Q}$ has two mirrors functions $\mathcal{Q}.mirror_1$ and $\mathcal{Q}.mirror_2$. Let p be an agent. By definition 8.1, $mirror_1(p)$ and $mirror_2(p)$ are either both defined or both undefined. If they are both defined, then

$$p \parallel mirror_1(p) \in G \wedge p \parallel mirror_2(p) \in G$$

By theorem 8.15

$$\Rightarrow \quad mirror_1(p) \preceq mirror_2(p) \wedge mirror_2(p) \preceq mirror_1(p)$$

by corollary 5.4

$$\Rightarrow \quad mirror_1(p) = mirror_2(p)$$

Since p was arbitrary, then $\mathcal{Q}.mirror_1 = \mathcal{Q}.mirror_2$. □

Perfectly reasonable agent algebras may fail to have a mirror function. The characterization of theorem 8.18 tells us that this may occur for the following two reasons:

- the parallel composition operator is unable to characterize the order of the algebra, i.e. the algebra does not have a conformance order relative to composition, or
- the compatibility set fails to have a greatest element.

In both cases the lack of a mirror function is due to insufficient information in the agent model. The following examples show that by extending the model it is possible to recover a mirror function and a conformance order.

Example 8.20 (Alphabet Algebra). Consider the agent algebra $\mathcal{Q}$ described in example 7.19, and let $G = 2^{\mathcal{A}} - \mathcal{A}$. Recall that $\mathcal{Q}$ has a G-conformance order relative to composition. We now show that $\mathcal{Q}$ has no mirror function relative to G. To do so, we consider the compatibility set of each agent, and then apply theorem 8.18.

Let p be an agent. It is easy to see that the set of agents compatible with p is

$$cmp(p) = \{q : \exists a[a \notin q \wedge a \notin p]\}.$$

The maximal elements of the compatibility set are therefore

$$maxcmp(p) = \{\mathcal{A} - \{a\} : a \notin p\}.$$

Observe that $maxcmp(p)$ is a set of incomparable agents. Thus $cmp(p)$ does not have a greatest element, and therefore, by theorem 8.18, $\mathcal{Q}$ does not have a mirror function relative to G. Because G is the only set of agents that characterizes the order relative to composition, $\mathcal{Q}$ has no mirror function relative to any G.

Example 8.21 (Locked Alphabet Algebra). In this example we present an extension of example 8.20 and we show that by adding extra information to the model it is possible to characterize the order with a mirror function.

The locked alphabet algebra $\mathcal{Q}$ is defined as follows:

- Agents are of the form $p = (A, L)$ where A and L are disjoint subsets of $\mathcal{Q}.\mathcal{A}$. The alphabet of p is $\alpha(p) = A \cup L$.
- $rename(r)(p)$ is defined whenever $\alpha(p) \subseteq dom(r)$. In that case $rename(r)(p) = (r(A), r(L))$, where r is naturally extended to sets.
- $proj(B)(p) = (A \cap B, L \cap B)$.
- $p_1 \parallel p_2$ is defined whenever $L_1 \cap L_2 = \emptyset$, $A_1 \cap L_2 = \emptyset$ and $A_2 \cap L_1 = \emptyset$. In that case

$$p_1 \parallel p_2 = (A_1 \cup A_2, L_1 \cup L_2).$$

The additional set of signals L is used by an agent p to indicate that no agent q can compose with p if q uses signals in L.

Theorem 8.22. Let $\preceq$ be an order for $\mathcal{Q}$ such that *rename*, *proj* and $\parallel$ are T-monotonic. Let $p = (A, L)$ and $p' = (A', L')$ be two agents. Then $p \preceq p'$ only if $A \subseteq A' \cup L'$ and $L \subseteq L'$.

Proof: Consider the agent $q = (A', \mathcal{A} - (A' \cup L'))$. Clearly, $p' \| q$ is defined, since $L' \cap \mathcal{A} - (A' \cup L') = \emptyset$ and $A' \cap L' = \emptyset$. Therefore, since $\parallel$ is T-monotonic and $p \preceq p'$, also $p \parallel q$ is defined. Hence:

$$\begin{aligned}
& L \cap \mathcal{A} - (A' \cup L') = \emptyset \wedge L \cap A' = \emptyset \wedge A \cap \mathcal{A} - (A' \cup L') = \emptyset \\
\Rightarrow\ & L \subseteq A' \cup L' \wedge L \cap A' = \emptyset \wedge A \subseteq A' \cup L' \\
\Rightarrow\ & A \subseteq A' \cup L' \wedge L \subseteq L'.
\end{aligned}$$

The requirements of *rename* and *proj* are subsumed by those of $\parallel$. □

We will consider the order such that $p \preceq p'$ if and only if $A \subseteq A' \cup L'$ and $L \subseteq L'$. The proof that the operators are T-monotonic is left to the reader.

Note that the subset of agents $P = \{(A, L) : L = \emptyset\}$ is closed under the operations and thus constitutes a subalgebra $\mathcal{P}$ of $\mathcal{Q}$. It is easy to show that $\mathcal{P}$ is isomorphic to the Alphabet Algebra of example 8.20. By extension, we consider the Locked Alphabet Algebra a superalgebra of the Alphabet Algebra.

The order can be characterized as a G-conformance order relative to composition where $G = \mathcal{D}$ includes all the agents of the algebra. Clearly G is downward closed relative to $\preceq$.

Let now $p' = (A', L')$ be an agent, and consider the set of agents $q = (A, L)$ that are compatible with p'. Since G is the set of all agents, an agent q is compatible with p' if and only if $q \parallel p'$ is defined, that is

$$L \cap L' = \emptyset \wedge A \cap L' = \emptyset \wedge A' \cap L = \emptyset$$

which translates to

$$cmp(p') = \{(A, L) : A \cap L = \emptyset \wedge A \subseteq \mathcal{A} - L' \wedge L \subseteq \mathcal{A} - (A' \cup L')\}.$$

Note that if $A \subseteq \mathcal{A} - L'$ and $L \subseteq \mathcal{A} - (A' \cup L')$, then $A \subseteq A' \cup (\mathcal{A} - (A' \cup L'))$ and $L \subseteq \mathcal{A} - (A' \cup L')$. Therefore, the agent $q = (A', \mathcal{A} - (A' \cup L')$ is the greatest element of $cmp(p')$.

Theorem 8.23. Let $p = (A, L)$ and $p' = (A', L')$ be two agents. Then $p \preceq p'$ if and only if $p \| (A', \mathcal{A} - (A' \cup L'))$ is defined.

Therefore $mirror(p') = (A', \mathcal{A} - (A' \cup L'))$ is a mirror function relative to G, and $\mathcal{Q}$ has a G-conformance order relative to composition.

Example 8.24 (IO Agent Algebra). Consider the IO agent algebra $\mathcal{Q}$ described in example 7.23 and let G be the set of agents that have no inputs. Then $\mathcal{Q}$ has a G-conformance order relative to composition. We now show that $\mathcal{Q}$ has no mirror function relative to G. Let $p' = (I', O')$ be an agent. The set of agents compatible with p' is

$$cmp(p') = \{q = (I_q, O_q) : I_q \subseteq O' \wedge I' \subseteq O_q \subseteq \mathcal{A} - O'\}.$$

The maximal elements of the compatibility set are therefore

$$maxcmp(p') = \{q = (I_q, O_q) : I_q = O' \wedge I' \subseteq O_q \subseteq \mathcal{A} - O'\}.$$

Since the agents in $maxcmp(p')$ are incomparable, $cmp(p')$ does not have a greatest element, and therefore, by theorem 8.18, $\mathcal{Q}$ does not have a mirror function relative to G.

Notice how every maximal element imposes a particular constraint for an agent to refine another. Let $p = (I, O)$ and $p' = (I', O')$ be two agents. Then the maximal element $q_1 = (O', I')$ characterizes an order (which is not T-monotonic) such that

$$p \preceq p' \Leftrightarrow I \subseteq I' \wedge O \supseteq O' \wedge O \cap I' = \emptyset.$$

On the other hand, the maximal element $q_2 = (O', \mathcal{A} - O')$ charaterizes the different order (again not T-monotonic) such that

$$p \preceq p' \Leftrightarrow I \subseteq \mathcal{A} - O' \wedge O \supseteq O' \wedge O \subseteq O'.$$

In other words, q_1 provides the constraint on the inputs, while q_2 constrains the outputs. Note that in this case these two maximal elements are sufficient to characterize the order, which is equal to the intersection of the two orders described.

Example 8.25 (Locked IO Agent Algebra). In this example we present an extension of example 8.24 and we show that by adding extra information to the model it is possible to characterize the order with a mirror function.

The locked IO Agent algebra $\mathcal{Q}$ is defined as follows:

- Agents are of the form $p = (I, O, L)$ where I, O and L are disjoint subsets of $\mathcal{Q}.\mathcal{A}$. The alphabet of p is $\alpha(p) = I \cup O \cup L$.
- *rename*$(r)(p)$ is defined whenever $\alpha(p) \subseteq dom(r)$. In that case *rename*$(r)(p) = (r(I), r(O), r(L))$, where r is naturally extended to sets.
- *proj*$(B)(p)$ is defined whenever $I \subseteq B$. In that case, *proj*$(B)(p) = (I, O \cap B, L \cap B)$.
- $p_1 \parallel p_2$ is defined whenever $(O_1 \cup L_1) \cap (O_2 \cup L_2) = \emptyset$, $I_1 \cap L_2 = \emptyset$ and $I_2 \cap L_1 = \emptyset$. In that case

$$p_1 \parallel p_2 = ((I_1 \cup I_2) - (O_1 \cup O_2), O_1 \cup O_2, L_1 \cup L_2).$$

The additional set of signals L is used by an agent p to indicate that no agent q can compose with p if q uses signals in L.

Theorem 8.26. Let $\preceq$ be an order for $\mathcal{Q}$ such that *rename*, *proj* and $\parallel$ are $\top$-monotonic. Then $p \preceq p'$ only if $I \subseteq I'$, $O' \subseteq O \subseteq O' \cup L'$ and $L \subseteq L'$.

Proof: The proof is similar to the proof of theorem 5.11. Let $p = (I, O, L)$ and $p' = (I', O', L')$ be two agents such that $p \preceq p'$. Then consider the agent $q = (O', I', \mathcal{A} - (I' \cup O' \cup L'))$ and deduce the conditions for which *rename*$(r)(p)$, *proj*$(B)(p)$ and $p \parallel q$ are all defined. □

We will consider the order such that $p \preceq p'$ exactly when $I \subseteq I'$, $O' \subseteq O \subseteq O' \cup L'$ and $L \subseteq L'$.

Theorem 8.27. The functions *rename*, *proj* and $\parallel$ are $\top$-monotonic with respect to $\preceq$.

Proof: The proof is similar to the proof of theorem 5.12. □

Note that the subset of agents $P = \{(I, O, L) : L = \emptyset\}$ is closed under the operations and thus constitutes a subalgebra $\mathcal{P}$ of $\mathcal{Q}$. It is easy to show that $\mathcal{P}$ is isomorphic to the IO Agent Algebra of example 8.24. By extension, we consider the Locked IO Agent Algebra a superalgebra of the IO Agent Algebra.

The order can be characterized as a G-conformance order relative to composition, where $G = \{(\emptyset, O, L)\}$ includes all the only the agents with no inputs. Clearly G is downward closed relative to $\preceq$.

Let now $p' = (I', O', L')$ be an agent, and consider the set of agents $q = (I, O, L)$ compatible with p'. We have

$$q \parallel p' = ((I \cup I') - (O \cup O'), O \cup O', L \cup L'),$$

with the following conditions for membership in G and for definedness:

$$\begin{aligned}
&I \cup I \subseteq O \cup O', \\
&(O \cup L) \cap (O' \cup L') = \emptyset, \\
&I \cap L' = \emptyset \wedge L \cap I' = \emptyset \wedge L \cap L' = \emptyset.
\end{aligned}$$

These conditions imply (since also for each agent, I, O and L must be disjoint) that

$$\begin{aligned}
\emptyset \subseteq \; & I \subseteq O' \\
I' \subseteq \; & O \subseteq \mathcal{A} - (O' \cup L') \\
\emptyset \subseteq \; & L \subseteq \mathcal{A} - (O \cup O' \cup L')
\end{aligned}$$

Notice that the two agents $q_1 = (I, O \cup \{a\}, L)$, and $q_2 = (I, O, L \cup \{a\})$ are comparable and $q_1 \preceq q_2$. Therefore the set of compatible agents of p' has a greatest element. It is easy to show that the greatest element is also a mirror function, so that

$$mirror(p') = (O', I', \mathcal{A} - (I' \cup O' \cup L')).$$

Hence, the algebra also has a G-conformance order relative to composition.

In the particular case of the simple IO agents, p' is of the form $p' = (I', O', \emptyset)$. Hence $mirror(p') = (O', I', \mathcal{A} - (I' \cup O'))$. Note how all the maximal elements found in example 8.24 are contained in $mirror(p')$ in the superalgebra. In the superalgebra, however, the compatibility set is extended upwards by agents that converge to a unique greatest element.

Example 8.28 (Dill's IO Agent Algebra). We have seen in example 7.29 that the Dill's IO Agent Algebra does not have a characterization in terms of conformance relative to composition. It is therefore impossible to find a mirror function in this case. We will however reconsider this example when we restrict the order to agents that share the same alphabet, below.

In this section we have seen examples of agent algebras that don't have a mirror function, despite having a G-conformance order relative to composition (see example 8.20 and example 8.24). The solution adopted in those cases consists of augmenting the model with enough information to let a single environment characterize the order. In the next two sections we explore alternative solutions that consist of adding some extra condition to the definition of the mirror function in order to restrict the size of the compatibility set.

8.1 Mirrors with Predicate

Let $\mathcal{Q}$ be an ordered agent algebra, and let p' be an agent. If $\mathcal{Q}$ has a G-conformance order relative to composition, then the compatibility set $cmp(p')$ of p' completely characterizes the set of agents p such that $p \preceq p'$ (see lemma 8.13). Each individual agent q in the compatibility set contributes to the characterization of the order by discriminating among two sets: the set of agents p that are compatible with q *do not conform* to p'; and the set of agents p that are compatible with q that *potentially* conform to p'. In other words, each compatible agent has a particular view of the conformance order.

When a mirror function exists, one agent (the greatest element) has an exact view of the conformance order. In that case, the compatibility set of p' is equal to the set of agents that conform to $mirror(p')$, and, viceversa, the compatibility set of $mirror(p')$ is equal to the set of agents that conform to p'. For an arbitrary element of the compatibility set we can only establish a containment relatioship.

Definition 8.29 (Refinement Set). Let $\mathcal{Q}$ be an ordered agent algebra, and let $p' \in \mathcal{Q}.D$. The *refinement set* of p', written $ref(p')$, is the set of agents p such that $p \preceq p'$:

$$ref(p') = \{p : p \preceq p'\}.$$

Lemma 8.30. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. Let p' be an agent such that $mirror(p')$ is defined. Then

$$ref(p') = cmp(mirror(p')).$$

Proof: The proof consists of the following series of double implications:

$p \in \mathit{ref}(p')$

by definition 8.29

$\Leftrightarrow \quad p \preceq p'$

by definition 8.1, since $\mathit{mirror}(p')$ is defined

$\Leftrightarrow \quad p \parallel \mathit{mirror}(p') \in G$

by definition 8.11

$\Leftrightarrow \quad p \in \mathit{cmp}(\mathit{mirror}(p'))$.

□

Lemma 8.31. Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Let p' be an agent and let $q \in \mathit{cmp}(p')$ be a compatible agent. Then

$$\mathit{ref}(p') \subseteq \mathit{cmp}(q).$$

Proof: The proof consists of the following series of implications:

$p \in \mathit{ref}(p')$

by definition 8.29

$\Leftrightarrow \quad p \preceq p'$

since $\mathcal{Q}$ has a G-conformance order relative to composition

$\Leftrightarrow \quad \forall q, p' \parallel q \in G \Rightarrow p \parallel q \in G$

since $q \in \mathit{cmp}(p')$, $q \parallel p' \in G$, therefore

$\Rightarrow \quad p \parallel q \in G$

by definition 8.11

$\Leftrightarrow \quad p \in \mathit{cmp}(q)$.

□

These two results are represented graphically in figure 1 and figure 2. Given an agent q in the compatibility set of p', we call the *discrimination set* of q the set of agents that q discriminates exactly for the purpose of conformance to p'.

Definition 8.32 (Discrimination Set). Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Let p' be an agent and let $q \in \mathit{cmp}(p')$ be a compatible agent. The *discrimination set* of q over p' is the set

$$\mathit{dis}_{p'}(q) = \{p : p \preceq p' \Leftrightarrow p \parallel q \in G\}.$$

Lemma 8.33 (Discrimination). Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Let p' be an agent and let $q \in \mathit{cmp}(p')$ be a compatible agent. Then

$$\mathit{dis}_{p'}(q) = (\mathcal{Q}.D - \mathit{cmp}(q)) \cup \mathit{ref}(p').$$

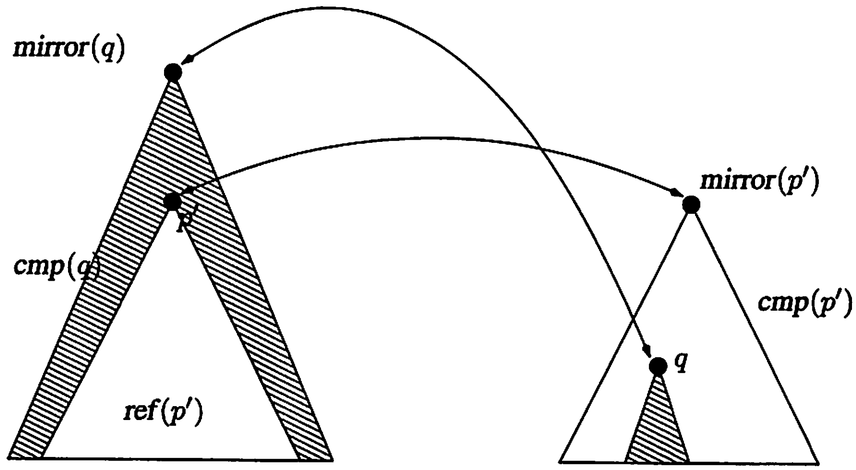

Figure 1: Refinement sets and compatibility sets with mirrors

Proof: We must show that $p \in (\mathcal{Q}.D - cmp(q)) \cup ref(p')$ if and only if $p \preceq p' \Leftrightarrow p \parallel q \in G$. For the forward direction we consider the following two cases:

- If $p \in \mathcal{Q}.D - cmp(q)$, then $p \notin cmp(q)$ and $p \parallel q \notin G$. By lemma 8.31, $ref(p') \subseteq cmp(q)$, therefore $p \notin ref(p')$. Therefore $p \not\preceq p'$. Hence $p \preceq p' \Leftrightarrow p \parallel q \in G$.
- If $p \in ref(p')$, then $p \preceq p'$. By lemma 8.31, $ref(p') \subseteq cmp(q)$, therefore $p \in cmp(q)$. Therefore $p \parallel q \in G$. Hence $p \preceq p' \Leftrightarrow p \parallel q \in G$.

For the reverse direction, let p be an agent such that $p \preceq p' \Leftrightarrow p \parallel q \in G$. We then consider the following two cases:

- If $p \preceq p'$, then $p \in ref(p')$ and therefore $p \in (\mathcal{Q}.D - cmp(q)) \cup ref(p')$.
- If $p \not\preceq p'$, then, by hypothesis, $p \parallel q \notin G$. Hence $p \notin cmp(q)$. Therefore $p \in \mathcal{Q}.D - cmp(q)$, and consequently $p \in (\mathcal{Q}.D - cmp(q)) \cup ref(p')$.

□

Corollary 8.34. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function relative to G. Let p' be an agent such that $mirror(p')$ is defined. Then

$$dis_{p'}(mirror(p')) = \mathcal{Q}.D.$$

Proof: The proof consists of the following equalities:

$$\begin{aligned}
dis_{p'}(mirror(p')) &= \{p : p \preceq p' \Leftrightarrow p \parallel mirror(p') \in G\} \\
&\quad \text{by lemma 8.33} \\
&= (\mathcal{Q}.D - cmp(mirror(p'))) \cup ref(p') \\
&\quad \text{by lemma 8.30} \\
&= (\mathcal{Q}.D - ref(p')) \cup ref(p') \\
&= \mathcal{Q}.D.
\end{aligned}$$

□

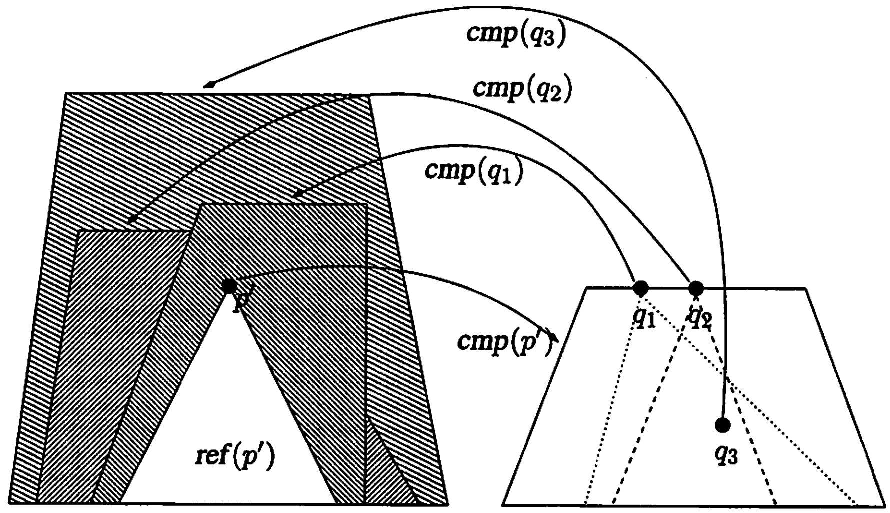

Figure 2: Compatibility sets of compatible agents

The above results show that every compatible agent can be used as a "mirror" if the characterization is restricted to its discrimination set. This suggests an extended notion of mirror function, whose validity is subject to the validity of a predicate.

Definition 8.35 (Mirror Function with Predicate). Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents of $\mathcal{Q}$. For each agent p', let $\mathit{pred}(p') \subseteq \mathcal{Q}.D$ be a predicate over $\mathcal{Q}.D$ such that $\mathit{ref}(p') \subseteq \mathit{pred}(p')$. Then, *$\mathcal{Q}$ has a mirror function with predicate relative to G* if and only if

1. $\mathcal{Q}.\mathit{mirror}$ (which we may simply write as "*mirror*" when there is no ambiguity about what agent algebra is being considered) is a partial function from D to D,
2. $\mathit{mirror}(p)$ is defined if and only if there exists q such that $p \parallel q \in G$,
3. If $p \in \mathit{pred}(p')$, then $p \preceq p'$ if and only if either $\mathit{mirror}(p')$ is undefined or $p \parallel \mathit{mirror}(p') \in G$.

Corollary 8.36. Let $\mathcal{Q}$ be an ordered agent algebra with a mirror function with predicate relative to G. Then, for all agents p and p', the following two statements are equivalent

1. $p \preceq p'$
2. $p \in \mathit{pred}(p')$ and either $\mathit{mirror}(p')$ is undefined or $p \parallel \mathit{mirror}(p') \in G$.

The regular mirror function can be interpreted as a mirror function with predicate by simply setting for all agents p'

$$\mathit{pred}(p') = \mathcal{Q}.D.$$

Hence mirror functions with predicate are more general than the regular mirror functions.

Unfortunately mirror functions with predicate do not enjoy the same characterization in terms of G-conformance relative to composition and greatest elements of the compatibility set (see theorem 8.18). A simple counterexample is obtained by considering the predicate

$$\mathit{pred}(p') = \{p : p \preceq p'\}.$$

In this case, any agent of the compatibility set can function as the mirror. This extreme case is, of course, useless, since the complexity of checking membership with the predicate is the same as the complexity of checking conformance. Mirror functions with predicate are therefore most useful when the predicate is relatively easy to check.

The choice of the predicate is guided by the following result.

Theorem 8.37. Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. For all agents p', let $mirror(p') \in cmp(p')$ be a compatible agent ($mirror(p')$ is undefined if $cmp(p') = \emptyset$), and let $pred(p') \subseteq \mathcal{Q}.D$ be a predicate. Then the following two conditions are equivalent:

1. $mirror$ is a mirror function with predicate $pred$ relative to G.
2. For all agents p', $ref(p') \subseteq pred(p')$ and if $mirror(p')$ is defined, $pred(p') \subseteq dis_{p'}(mirror(p'))$.

Proof: For the forward direction, by definition 8.35, for all agents p', $ref(p') \subseteq pred(p')$. Let now p' be an agent such that $mirror(p')$ is defined and let $p \in pred(p')$ be an agent. Then,

$p \in pred(p')$

by definition 8.35, since $\mathcal{Q}$ has a mirror function with predicate relative to G

$\Rightarrow \quad p \preceq p' \Leftrightarrow p \parallel mirror(p') \in G$

by definition 8.32

$\Rightarrow \quad p \in dis_{p'}(mirror(p'))$.

Hence, $pred(p') \subseteq dis_{p'}(mirror(p'))$.

For the reverse direction, assume $ref(p') \subseteq pred(p') \subseteq dis_{p'}(mirror(p'))$. Clearly $mirror$ is a partial function, and $mirror(p)$ is defined if and only if there exists q such that $p \parallel q \in G$. Let now p be an agent such that $p \in pred(p')$. We must show that $p \preceq p'$ if and only if either $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$. We consider two cases.

Assume $mirror(p')$ is undefined. Then, by hypothesis, $cmp(p') = \emptyset$, and therefore, since $\mathcal{Q}$ has a G-conformance order relative to composition, for all agents p, $p \preceq p'$. Therefore, $p \preceq p'$ if and only if $mirror(p')$ is undefined.

Conversely, assume $mirror(p')$ is defined. Then,

$p \in pred(p')$

since by hypothesis $pred(p') \subseteq dis_{p'}(mirror(p'))$

$\Rightarrow \quad p \in dis_{p'}(mirror(p'))$

by definition 8.32

$\Rightarrow \quad p \preceq p \Leftrightarrow p \parallel mirror(p') \in G$.

Hence $mirror$ is a mirror function with predicate $pred$ relative to G. □

The greater the element in the compatibility set, the larger the discrimination set.

Lemma 8.38. Let $\mathcal{Q}$ be an ordered agent algebra with a G-conformance order relative to composition. Let p' be an agent and let q_1 and q_2 be compatible agents such that $q_1 \preceq q_2$. Then

$$dis_{p'}(q_1) \subseteq dis_{p'}(q_2).$$

Since greater elements have larger discrimination sets, and since $pred(p')$ must be a subset of the discrimination set of $mirror(p')$, it is convenient to choose a maximal element of the compatibility set for $mirror(p')$. In this way, we have the maximum flexibility in choosing a predicate that is computationally easy to check. However, unlike regular mirror functions, the mirror of an agent with predicate is not necessarily a maximal element of the compatibility set.

The following examples show the use of mirror functions with predicate in the cases where a regular mirror function does not exist.

Example 8.39 (IO Agent Algebra). Example 8.24 shows that the IO agent algebra does not have a mirror function relative to G, despite having a G-conformance order relative to composition. In this example we show how to derive a mirror function with predicate.

Let $p' = (I', O')$ be an agent. As shown in example 8.24, the agent $q_1 = (O', I')$ is a maximal element of the compatibility set of p'. We now wish to use q_1 as a mirror of p' with the use of a predicate. To do so, we compute the discrimination set of q_1:

$$\begin{aligned} dis_{p'}(q_1) &= (\mathcal{Q}.D - cmp(q_1)) \cup ref(p') \\ &= (\mathcal{Q}.D - \{(I,O) : I \subseteq I' \wedge O' \subseteq O \subseteq \mathcal{A} - I'\}) \cup \{(I,O) : I \subseteq I' \wedge O = O'\} \\ &= \{(I,O) : I' \subsetneq I \vee O \subsetneq O' \vee \mathcal{A} - I' \subsetneq O\} \cup \{(I,O) : I \subseteq I' \wedge O = O'\} \\ &= \{(I,O) : I' \subsetneq I \vee O \subsetneq O' \vee \mathcal{A} - I' \subsetneq O \vee (I \subseteq I' \wedge O = O')\} \end{aligned}$$

Recall that $pred(p')$ must include $ref(p')$. A reasonable choice for $pred(p')$ in this case is the following:

$$pred(p') = \{p : \alpha(p) \subseteq \alpha(p')\}.$$

This predicate is easy to check for finite alphabets, and satisfies the condition $ref(p') \subseteq pred(p') \subseteq dis_{p'}(q_1)$. Therefore, by theorem 8.37, $\mathcal{Q}$ has a mirror function with predicate relative to G, where

$$mirror((I,O)) = (O,I), \quad pred(p') = \{p : \alpha(p) \subseteq \alpha(p')\}.$$

Note that q_1 is not the only compatible agent that can be used as a mirror with the above predicate. For example, the maximally compatible agent $q_2 = (O', \mathcal{A} - O')$ has the following discrimination set:

$$\begin{aligned} dis_{p'}(q_2) &= (\mathcal{Q}.D - cmp(q_2)) \cup ref(p') \\ &= (\mathcal{Q}.D - \{(I,O) : I \subseteq \mathcal{A} - O' \wedge O = O'\}) \cup \{(I,O) : I \subseteq I' \wedge O = O'\} \\ &= \{(I,O) : \mathcal{A} - O' \subset I \vee O \neq O'\} \cup \{(I,O) : I \subseteq I' \wedge O = O'\} \\ &= \{(I,O) : \mathcal{A} - O' \subset I \vee O \neq O' \vee (I \subseteq I' \wedge O = O')\} \end{aligned}$$

It is easy to check that the condition $ref(p') \subseteq pred(p') \subseteq dis_{p'}(q_2)$ is satisfied. Hence, $\mathcal{Q}$ has also the following mirror function with predicate:

$$mirror((I,O)) = (O, \mathcal{A} - O), \quad pred(p') = \{p : \alpha(p) \subseteq \alpha(p')\}.$$

We have noted how mirror functions with predicate lose the characterization in terms of conformance order that simple mirror functions have. For a restricted case, however, we can reduce a mirror function with predicate to a regular mirror function by extending the model as in example 8.21 and example 8.25. The construction consists of augmenting the model by providing each agent with the information conveyed by the predicate of their mirror. This construction is still somewhat preliminary, as it doesn't guarantee a downward closed conformance set. In addition, the conformance set does not necessarily enjoy the properties that are necessary for applying theorem 9.4 below.

Theorem 8.40. Let $\mathcal{Q}$ be an agent algebra with a mirror function with predicate relative to G, such that $pred(p) = pred(mirror(p))$. If $\overline{G}$ is downward closed, then the agent algebra $\overline{\mathcal{Q}}$ has a mirror function relative to $\overline{G}$, where

- $\overline{\mathcal{Q}}.D = \{(p, set) : p \in \mathcal{Q}.D \wedge set \subseteq \mathcal{Q}.D \wedge pred(p) \cap set = \emptyset\}$
- $(p, set) \preceq (p', set')$ if and only if either $\mathcal{Q}.mirror(p')$ is not defined, or, if defined,

  $set \subseteq set'$,
  $pred(p) \subseteq set' \cup pred(p')$ and
  $p \parallel \mathcal{Q}.mirror(p') \in G$.

- $proj(B)((p, set_p)) = (proj(B)(p), proj(B)(set_p))$ if all quantities are defined.
- $rename(r)((p, set_p)) = (rename(r)(p), rename(r)(set_p))$ if all quantities are defined.
- $(p, set_p) \parallel (q, set_q)$ is defined if and only if

  $p \parallel q$ is defined,
  $pred(p) \cap set_q = \emptyset$,
  $pred(q) \cap set_p = \emptyset$ and
  $set_p \cap set_q = \emptyset$.

  In that case

  $$(p, set_p) \parallel (q, set_q) = (p \parallel q, set_p \cup set_q)$$

- $\overline{\mathcal{Q}}.mirror((p, set))$ is defined if and only if $\mathcal{Q}.mirror(p)$ is defined. In that case, $\overline{\mathcal{Q}}.mirror((p, set)) = (\mathcal{Q}.mirror(p), \mathcal{Q}.D - (set \cup pred(p)))$
- $\overline{G} = G \times 2^{\mathcal{Q}.D}$

Proof: We must show that $\overline{\mathcal{Q}}.mirror$ is a mirror function relative to $\overline{G}$.

Clearly $\overline{\mathcal{Q}}.mirror$ is a partial function from $\overline{\mathcal{Q}}.D$ to $\overline{\mathcal{Q}}.D$. Also, if $\overline{\mathcal{Q}}.mirror((p, set))$ is defined, then $\overline{\mathcal{Q}}.mirror((p, set)) \parallel (p, set) \in \overline{G}$. Conversely, if $(q, set_q) \parallel (p, set) \in \overline{G}$, then $q \parallel p \in G$, hence $\mathcal{Q}.mirror(p)$ is defined, and therefore $\overline{\mathcal{Q}}.mirror((p, set))$ is defined.

Assume now that $(p, set) \preceq (p', set')$, and assume that $\overline{\mathcal{Q}}.mirror((p', set')) = (\mathcal{Q}.mirror(p'), \mathcal{Q}.D - (set' \cup pred(p')))$ is defined. Then

- By hypothesis $p \parallel \mathcal{Q}.mirror(p') \in G$.
- $pred(p) \cap (\mathcal{Q}.D - (set' \cup pred(p'))) = \emptyset$, since $pred(p) \subseteq set' \cup pred(p')$.
- $pred(\mathcal{Q}.mirror(p')) \cap set = \emptyset$, since by hypothesis $pred(\mathcal{Q}.mirror(p')) = pred(p')$, $pred(p') \cap set' = \emptyset$ and $set \subseteq set'$.
- $set \cap (\mathcal{Q}.D - (set' \cup pred(p'))) = \emptyset$, since $set \subseteq set'$.

Therefore $(p, set) \parallel (\mathcal{Q}.mirror(p'), \mathcal{Q}.D - (set' \cup pred(p'))) \in \overline{G}$.

Conversely, assume $(p, set) \parallel (\mathcal{Q}.mirror(p'), \mathcal{Q}.D - (set' \cup pred(p'))) \in \overline{G}$. Then

- $set \subseteq set' \cup pred(p')$, since $set \cap (\mathcal{Q}.D - (set' \cup pred(p'))) = \emptyset$. In addition, $set \cap pred(p') = \emptyset$, since $set \cap pred(\mathcal{Q}.mirror(p')) = \emptyset$ and $pred(\mathcal{Q}.mirror(p')) = pred(p')$. Therefore, $set \subseteq set'$.
- $pred(p) \subseteq set' \cup pred(p')$, since $pred(p) \cap (\mathcal{Q}.D - (set' \cup pred(p'))) = \emptyset$.
- By hypothesis $p \parallel \mathcal{Q}.mirror(p') \in G$.

Therefore $(p, set) \preceq (p', set')$.

Similarly, if $\overline{\mathcal{Q}}.mirror((p', set'))$ is not defined, then $(p, set) \preceq (p', set')$ if and only if $\overline{\mathcal{Q}}.mirror((p', set'))$ is not defined.

Therefore, by definition 8.1, $\overline{\mathcal{Q}}.mirror$ is a mirror function for $\overline{\mathcal{Q}}$ relative to $\overline{G}$. □

Theorem 8.41. Let $\mathcal{Q}$ and $\overline{\mathcal{Q}}$ be as in theorem 8.40. Then the function $e : \mathcal{Q}.D \mapsto \overline{\mathcal{Q}}.D$ such that for all agents p

$$e(p) = (p, \emptyset)$$

is an embedding.

Proof: It is easy to show that e commutes with the operators of the algebra, that is, for example, that

$$proj(B)(e(p)) = e(proj(B)(p)).$$

To complete the proof we must show that $p \preceq p'$ if and only if $(p, \emptyset) \preceq (p', \emptyset)$.

Let p and p' be such that $p \preceq p'$. If $\mathcal{Q}.mirror(p')$ is not defined, then $(p, \emptyset) \preceq (p', \emptyset)$. Alternatively, assume $\mathcal{Q}.mirror(p')$ is defined. Then, since $p \preceq p'$, $p \in pred(p')$ and $p \parallel \mathcal{Q}.mirror(p') \in G$. Since $pred$ is monotonic relative to $\preceq$, $pred(p) \subseteq pred(p')$. Therefore, by definition of $\overline{\mathcal{Q}}$, $(p, \emptyset) \preceq (p', \emptyset)$.

Conversely, assume $(p, \emptyset) \preceq (p', \emptyset)$. If $\mathcal{Q}.mirror(p')$ is not defined then $p \preceq p'$. Alternatively, assume $\mathcal{Q}.mirror(p')$ is defined. Then, by definition of $\overline{\mathcal{Q}}$, $pred(p) \subseteq pred(p')$, and therefore, since $p \in pred(p)$, $p \in pred(p')$. In addition, $p \parallel \mathcal{Q}.mirror(p') \in G$. Therefore, by definition 8.35, also $p \preceq p'$.

Hence, $p \preceq p'$ if and only if $(p, \emptyset) \preceq (p', \emptyset)$. □

Corollary 8.42. Let $\mathcal{Q}$ and $\overline{\mathcal{Q}}$ be as in theorem 8.40 such that $\overline{G}$ is downward closed, and let $(p', \emptyset)$ be an agent of $\overline{\mathcal{Q}}$. If $(p, set) \preceq (p', \emptyset)$, then $set = \emptyset$.

The above results show that if p' is an agent in $\mathcal{Q}$, then the mirror of $(p', \emptyset)$ in $\overline{\mathcal{Q}}$ characterizes exactly the agents p such that $p \preceq p'$.

8.2 Mirrors in Subalgebras

In the previous section we have employed a predicate to focus the application of the mirror function to only those agents that the mirror can discriminate. Here we use an alternative approach, and consider only a subset of the agents to reduce the size of the compatibility sets. We choose the subset so that it is downward closed, and closed under parallel composition, thus effectively constructing a subalgebra when the operators of projection and renaming are removed from the signature. Since the compatibility sets are smaller, subalgebras have a greater chance to have a G-conformance order relative to composition and a mirror function.

An example that is particularly useful in practice is the subset of agents that have the same alphabet. In particular, we are interested in studying the conformance order and the corresponding mirror function for algebras whose order satisfies the constraint that $p \preceq p'$ only if $\alpha(p) = \alpha(p')$. Note that if $\alpha(p) = \alpha(q)$, then $\alpha(p \parallel q) = \alpha(p) = \alpha(q)$. Therefore the subset of agents with a certain alphabet that satisfy the above constraint consitute a subalgebra of the original agent algebra (restricted to parallel composition only). Note also that the projection and renaming operators have no effect in determining conformance relative to composition and mirror functions. Therefore, the results of the previous sections apply to this restricted case, provided that the necessary restrictions on the alphabet are enforced throughout.

Let $\mathcal{Q}$ be an agent algebra and assume that for all alphabets A, the algebra $\mathcal{P}$ such that $\mathcal{P}.D = \{p : \alpha(p) = A\}$ is a subalgebra of $\mathcal{Q}$. Assume also that each subalgebra has a G-conformance order relative to composition and a mirror function relative to G. Note that since $\mathcal{P}.D$ must be downward closed for all alphabets, $p \preceq p'$ only if $\alpha(p) = \alpha(p')$. The results obtained in the subalgebras can be rephrased in terms of the original algebra by restricting the definitions of conformance order and mirror function to apply only when the alphabets of the agents involved are the same. Note that we are *not* changing the definition of conformance, but we are simply reflecting the restrictions of the subalgebra in the superalgebra.

Definition 8.43. Let $\mathcal{Q}$ be an agent algebra and let G be a downward closed set of agents of $\mathcal{Q}$. $\mathcal{Q}$ has a *same alphabet G-conformance order relative to composition* if and only if for all agents p and p', $p \preceq p'$ if and only if $\alpha(p) = \alpha(p')$ and for all agents q such that $\alpha(q) = \alpha(p')$, if $p' \parallel q \in G$ then $p \parallel q \in G$.

Definition 8.44. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents of $\mathcal{Q}$. Then, *$\mathcal{Q}$ has a same alphabet mirror function relative to G* if and only if

1. $\mathcal{Q}.mirror$ is a partial function from D to D,
2. $mirror(p)$ is defined if and only if there exists q such that $\alpha(q) = \alpha(p)$ and $p \parallel q \in G$,
3. $p \preceq p'$ if and only if $\alpha(p) = \alpha(p')$ and either $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$.

The additional conditions in these definitions consistently restrict the alphabets of the agents involved to the alphabet of the agent for which we are considering the mirror.

In particular we are interested in the characterization of the mirror in terms of the greatest element of the compatibility set and of conformance relative to composition.

Definition 8.45 (Compatibility Set). Let $\mathcal{Q}$ be an ordered agent algebra and G a downward closed set of agents. The *alphabet invariant G-compatibility set* of an agent p, written $cmp(p)$, is defined as follows:

$$cmp(p) = \{q : \alpha(q) = \alpha(p) \wedge p \parallel q \in G\}$$

Theorem 8.46. Let $\mathcal{Q}$ be an ordered agent algebra and let G be a downward closed set of agents. Then the following two statement are equivalent:

1. $\mathcal{Q}$ has an alphabet invariant mirror function relative to G.
2. $\mathcal{Q}$ has an alphabet invariant G-conformance order relative to composition, and for all agents p', $cmp(p')$ is either empty or if it is not empty it has a greatest element.

8.3 Construction of Algebras

In this section we explore conformance and mirrors for the direct product of algebras and for subalgebras.

Theorem 8.47. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras with a G_1 and G_2-conformance order, respectively. Let $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ be the direct product (definition 4.1) of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ and let $G = G_1 \times G_2$. Then for all $p, p' \in \mathcal{Q}.D$, if $p \preceq_{\mathcal{Q}} p'$ then for all expression contexts E, if $E[p'] \in G$ then $E[p] \in G$.

Proof: Let $p = \langle p_1, p_2 \rangle$ and $p' = \langle p'_1, p'_2 \rangle$ be agents such that $p \preceq p'$. The proof consists of the following serier of implications:

$$
\begin{aligned}
& p \preceq p' \\
& \quad \text{by definition 4.1} \\
& \Leftrightarrow \quad p_1 \preceq_{\mathcal{Q}_1} p'_1 \wedge p_2 \preceq_{\mathcal{Q}_2} p'_2 \\
& \quad \text{by hypothesis} \\
& \Leftrightarrow \quad (\forall E, E[p'_1] \in G_1 \Rightarrow E[p_1] \in G_1) \wedge (\forall E, E[p'_2] \in G_2 \Rightarrow E[p_2] \in G_2) \\
& \quad \text{by definition 4.1} \\
& \Rightarrow \quad \forall E, E[p'] \in G \Rightarrow E[p] \in G.
\end{aligned}
$$

□

Unfortunately the reverse of the last implication in the proof above does not hold, that is $\mathcal{Q}$ does not necessarily have a G-conformance order. This is because a context E may be defined for an agent p_1, while it is not defined for the pair $\langle p_1, p_2 \rangle$. However, the result holds in the presence of mirror functions, when the mirror function is always defined. In that case, in fact, the expression contexts can be reduced to a single environment, and the difficulty above disappears.

Theorem 8.48. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be agent algebras with a mirror function relative to G_1 and G_2, respectively, such that for all agents p, $mirror(p)$ is defined. Let $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ be the direct product (definition 4.1) of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ and let $G = G_1 \times G_2$. Then for all agents $\langle p_1, p_2 \rangle \in \mathcal{Q}.D$, $mirror(\langle p_1, p_2 \rangle) = \langle mirror(p_1), mirror(p_2) \rangle$ is a mirror function for $\mathcal{Q}$ relative to G.

Proof: Clearly $\mathcal{Q}.mirror$ is a partial (in fact, total) function. Since $mirror$ is always defined, we must show that for all $p = \langle p_1, p_2 \rangle$ there exists q such that $p \parallel q \in G$.

$$
\begin{aligned}
& mirror(\langle p_1, p_2 \rangle)\downarrow \\
& \quad \text{by hypothesis} \\
& \Leftrightarrow \quad mirror(p_1)\downarrow \wedge mirror(p_2)\downarrow \\
& \quad \text{by definition 8.1} \\
& \Leftrightarrow \quad (\exists q_1, p_1 \parallel q_1 \in G_1) \wedge (\exists q_2, p_2 \parallel q_2 \in G_2) \\
& \quad \text{by definition 4.1} \\
& \Leftrightarrow \quad \exists \langle q_1, q_2 \rangle, \langle p_1, p_2 \rangle \parallel \langle q_1, q_2 \rangle \in G \\
& \Leftrightarrow \quad \exists q, p \parallel q \in G
\end{aligned}
$$

It remains to show that for all $p, p' \in \mathcal{Q}.D$, $p \preceq p'$ if and only if $p \parallel mirror(p') \in G$. Let $p = \langle p_1, p_2 \rangle$ and $p' = \langle p'_1, p'_2 \rangle$.

$$
\begin{aligned}
& p \preceq p' \\
& \Leftrightarrow \quad \langle p_1, p_2 \rangle \preceq \langle p'_1, p'_2 \rangle \\
& \quad \text{by definition 4.1} \\
& \Leftrightarrow \quad p_1 \preceq p'_1 \wedge p_2 \preceq p'_2 \\
& \quad \text{by definition 8.1, since } mirror(p'_1) \text{ and } mirror(p'_2) \text{ are both defined} \\
& \Leftrightarrow \quad p_1 \parallel mirror(p'_1) \in G_1 \wedge p_2 \parallel mirror(p'_2) \in G_2 \\
& \quad \text{by definition 4.1} \\
& \Leftrightarrow \quad \langle p_1, p_2 \rangle \parallel \langle mirror(p'_1), mirror(p'_2) \rangle \in G \\
& \quad \text{by hypothesis} \\
& \Leftrightarrow \quad \langle p_1, p_2 \rangle \parallel mirror(\langle p'_1, p'_2 \rangle) \in G \\
& \Leftrightarrow \quad p \parallel mirror(p') \in G
\end{aligned}
$$

□

A subalgebra may fail to preserve a G-conformance order, but is well behaved when it is closed under a mirror function, if it exists.

Theorem 8.49. Let $\mathcal{Q}'$ be an ordered agent algebra with a G'-conformance order and let $\mathcal{Q}$ be a subalgebra of $\mathcal{Q}'$. Let $G = G' \cap \mathcal{Q}.D$. Then for all agents p and p' in $\mathcal{Q}$, if $p \preceq_{\mathcal{Q}} p'$ then for all expression contexts E over $\mathcal{Q}$, if $E[p'] \in G$, then $E[p] \in G$.

Proof: Let $\mathcal{E}'$ be the set of expressions over $\mathcal{Q}'$, and let $\mathcal{E}$ be the set of expressions over $\mathcal{Q}$. Note that since $\mathcal{Q}$ is a subalgebra of $\mathcal{Q}'$, an expression over $\mathcal{Q}$ is also an expression over $\mathcal{Q}'$, and therefore $\mathcal{E} \subseteq \mathcal{E}'$.

Let now p and p' be elements of $\mathcal{Q}.D$. The proof consists of the following series of implications.

$$
\begin{aligned}
& p \preceq_{\mathcal{Q}} p' \\
& \quad \text{by definition 5.22} \\
& \Leftrightarrow \quad p \preceq_{\mathcal{Q}'} p' \\
& \quad \text{since } \mathcal{Q}' \text{ has a } G'\text{-conformance order, by definition 7.3} \\
& \Leftrightarrow \quad \forall E \in \mathcal{E}', E[p'] \in G' \Rightarrow E[p] \in G' \\
& \quad \text{since } \mathcal{Q}.D \text{ is closed in } \mathcal{Q}'.D \text{ under the operators, } G = G' \cap \mathcal{Q}.D, \mathcal{E} \subseteq \mathcal{E}', \\
& \quad \text{and since for all } p \in \mathcal{Q}.D,\ E[p'] \in G' \Leftrightarrow E[p'] \in G, \\
& \Rightarrow \quad \forall E \in \mathcal{E}, E[p'] \in G \Rightarrow E[p] \in G
\end{aligned}
$$

□

The reverse of the last implication does not hold. In fact, while it is true that if $E[p] \in G$, then $E[p] \in G'$, the subalgebra can only consider a subset of the contexts, and may therefore be unable completely characterize the order.

If however $\mathcal{Q}$ is a subalgebra of $\mathcal{Q}'$ in the sense of definition 4.6 (i.e. without considering the order), and the subalgebra does have a G-conformance order, then the orders are related by the following result.

Theorem 8.50. Let $\mathcal{Q}'$ be an ordered agent algebra with a G'-conformance order and let $\mathcal{Q}$ be a subalgebra of $\mathcal{Q}'$. Let $G = G' \cap \mathcal{Q}.D$ and assume $\mathcal{Q}$ has a G-conformance order. Then for all agents p and p' in $\mathcal{Q}$,

$$p \preceq_{Q'} p' \Rightarrow p \preceq_{Q} p'.$$

Proof: Let now p and p' be elements of $\mathcal{Q}.D$. The proof consists of the following series of implications.

$p \preceq_{Q'} p'$

since $\mathcal{Q}'$ has a G'-conformance order, by definition 7.3

$\Leftrightarrow \quad \forall E \in \mathcal{E}', E[p'] \in G' \Rightarrow E[p] \in G'$

since $\mathcal{Q}.D$ is closed in $\mathcal{Q}'.D$ under the operators, $G = G' \cap \mathcal{Q}.D$, $\mathcal{E} \subseteq \mathcal{E}'$,

and since for all $p \in \mathcal{Q}.D$, $E[p'] \in G' \Leftrightarrow E[p'] \in G$,

$\Rightarrow \quad \forall E \in \mathcal{E}, E[p'] \in G \Rightarrow E[p] \in G$

since $\mathcal{Q}$ has a G-conformance order, by definition 7.3

$\Leftrightarrow \quad p \preceq_{Q} p'$

□

Theorem 8.51. Let $\mathcal{Q}'$ be an ordered agent algebra with a mirror function *mirror* relative to G' and let $\mathcal{Q}$ be a subalgebra of $\mathcal{Q}'$ closed under *mirror*. Let $G = G' \cap \mathcal{Q}.D$. Then $\mathcal{Q}$ has a mirror function relative to G.

Proof: We show that that *mirror* is a mirror function for $\mathcal{Q}$ relative to G. Clearly *mirror* is a partial function.

Let now $p \in \mathcal{Q}.D$ be an agent. If $mirror(p)$ is defined, then, since $\mathcal{Q}$ is closed under mirror, $mirror(p) \in \mathcal{Q}.D$. Since *mirror* is a mirror function relative to G' for $\mathcal{Q}'$, by lemma 8.2, $p \parallel mirror(p) \in G'$. Since $p \in \mathcal{Q}.D$, $mirror(p) \in \mathcal{Q}.D$, and since $\mathcal{Q}$ is closed under parallel composition, $p \parallel mirror(p) \in \mathcal{Q}.D$. Therefore, since $G = G' \cap \mathcal{Q}.D$, $p \parallel mirror(p) \in G$. Hence, if $mirror(p)$ is defined, then there exists q (i.e. $mirror(p)$) such that $p \parallel q \in G$.

Conversely, if there exists $q \in \mathcal{Q}.D$ such that $p \parallel q \in G$, then, since $\mathcal{Q} \subseteq \mathcal{Q}'$, also $p \parallel q \in G'$. Therefore, by definition 8.1, $mirror(p)$ is defined.

Let now $p \in \mathcal{Q}.D$ and $p' \in \mathcal{Q}.D$. It remains to show that $p \preceq p'$ if and only if either $mirror(p')$ is undefined or $p \parallel mirror(p') \in G$.

$p \preceq_{Q} p'$

since $\mathcal{Q} \subseteq \mathcal{Q}'$

$\Leftrightarrow \quad p \preceq_{Q'} p'$

since $\mathcal{Q}'$ has a mirror function relative to G', by definition 8.1

$\Leftrightarrow \quad mirror(p')\uparrow \vee p \parallel mirror(p') \in G'$

since $\mathcal{Q}$ is closed under mirror and $G = G' \cap \mathcal{Q}.D$

$\Leftrightarrow \quad mirror(p')\uparrow \vee p \parallel mirror(p') \in G$

□

9 Local Specification Synthesis

With conformance we have addressed the problem of characterizing substitutability under all possible contexts. Relative conformance has been introduced to reduce (whenever possible) *the complexity* of the problem by considering only a limited set of contexts. Relative conformance, however, when applicable, does not change the notion of substitutability, since, in that case, relative and general conformance coincide (cfr. theorem 7.17).

In this section we address the problem of deriving the local specification for an agent in a context, such that when an agent that satisfies the local specification is substituted in the context, the resulting system satisfies a global specification. Instances of this problem include supervisory-control synthesis [1], the rectification and optimization problem [3], and protocol conversion [11]. We will show that, under certain conditions, a mirror function provides us with a closed form solution.

Definition 9.1 (Local Specification). Let $\mathcal{Q}$ be an ordered agent algebra, E an expression context, and let p' be an agent. A *local specification for p' in E* is an agent q such that for all agents p,

$$p \preceq q \Leftrightarrow E[p] \preceq p'.$$

In the rest of this section we address the problem of deriving the local specification q, given the expression context E and the global specification p'. The solution involves the use of the mirror function. However, to solve the equation for the local specification, the conformance set must have some additional closure properties. We call a conformance set with these additional properties a *rectification set.*

Definition 9.2 (Rectification Set). Let $\mathcal{Q}$ be an agent algebra. A set $G \subset \mathcal{Q}.D$ is a *rectification set* if it satisfies the following requirements:

Downward closure If $p' \in G$ and $p \preceq p'$, then $p \in G$.

Closure under projection If $p \in G$, then for all alphabets B, $proj(B)(p)$ is defined and $proj(B)(p) \in G$.

Closure under inverse projection If $p \in G$, then for all alphabets B and all agents p', if $proj(B)(p') = p$ then $p' \in G$.

Closure under renaming If $p \in G$, then for all bijections r, if $rename(r)(p)$ is defined then $rename(r)(p) \in G$.

An agent algebra must be normalizable to synthesize a local specification. In fact, we need two additional properties to make sure that certain operations are well defined.

Definition 9.3 (Rectifiable Algebra). Let $\mathcal{Q}$ be a normalizable agent algebra. Then $\mathcal{Q}$ is rectifiable if it satisfies the following axioms, where p is an agent:

A26. $rename(r)(p)$ is defined if and only if $\alpha(p) \subseteq dom(r)$.

A27. For all alphabets A such that $\alpha(p) \subseteq A$, $rename(id_A)(p) = p$.

We can now state and prove the main result of this section.

Theorem 9.4 (Local Specification Synthesis). Let $\mathcal{Q}$ be an ordered rectifiable algebra and let G be a rectification set, such that $\mathcal{Q}$ has a G-conformance order relative to composition. Assume $\mathcal{Q}$ has a mirror function relative to G.

Let E be an expression context, and let p be an agent such that $mirror(p)$ is defined. Let

$$proj(B)(rename(r)(\beta) \parallel q)$$

be an expression in RCP normal form equivalent to E. Let $A_1 = codom(r) \cup \alpha(q) \cup B$ and let $\hat{r}^{-1}$ be an extension of r^{-1} to A_1 such that $\hat{r}^{-1}$ is a bijection. Then

$$E[\beta] \preceq p$$

if and only if

$$\beta \preceq mirror(rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p)))) \text{ and } \alpha(\beta) \subseteq dom(r).$$

Proof: The proof is composed of the following series of double implications.

$proj(B)(rename(r)(\beta) \parallel q) \preceq p$

by the characterization of "$\preceq$" in terms of G, since $mirror(p)$ exists

$\Leftrightarrow \quad proj(B)(rename(r)(\beta) \parallel q) \parallel mirror(p) \in G$

since G is closed under projection and inverse projection

$\Leftrightarrow \quad proj(B)(proj(B)(rename(r)(\beta) \parallel q) \parallel mirror(p)) \in G$

since, by A1, $\alpha(proj(B)(rename(r)(\beta) \parallel q)) \subseteq B$ and

since $\alpha(proj(B)(rename(r)(\beta) \parallel q) \cap \alpha(mirror(p)) \subseteq B$,

therefore by A25

$\Leftrightarrow \quad proj(B)(proj(B)(rename(r)(\beta) \parallel q)) \parallel proj(B)(mirror(p)) \in G$

by A20

$\Leftrightarrow \quad proj(B)(rename(r)(\beta) \parallel q) \parallel proj(B)(mirror(p)) \in G$

by A20

$\Leftrightarrow \quad proj(B)(rename(r)(\beta) \parallel q) \parallel proj(B)(proj(B)(mirror(p))) \in G$

since, by A1, $\alpha(proj(B)(mirror(p))) \subseteq B$ and

since $\alpha(rename(r)(\beta) \parallel q) \cap \alpha(proj(B)(mirror(p))) \subseteq B$,

therefore by A25

$\Leftrightarrow \quad proj(B)(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p))) \in G$

since G is closed under projection and inverse projection

$\Leftrightarrow \quad rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p)) \in G$

by A27

$\Leftrightarrow \quad rename(id_{A_1})(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p))) \in G$

since $\hat{r}^{-1}$ is a bijection over A_1

$\Leftrightarrow \quad rename(\hat{r} \circ \hat{r}^{-1})(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p))) \in G$

by A21

$\Leftrightarrow \quad rename(\hat{r})(rename(\hat{r}^{-1})(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p)))) \in G$

$\Leftrightarrow$ $rename(\hat{r})(rename(\hat{r}^{-1})(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p)))) \in G$
since G is closed under rename (and consequently under inverse rename)

$\Leftrightarrow$ $rename(\hat{r}^{-1})(rename(r)(\beta) \parallel q \parallel proj(B)(mirror(p))) \in G$
by A24

$\Leftrightarrow$ $rename(\hat{r}^{-1})(rename(r)(\beta)) \parallel rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p))) \in G$
by A21

$\Leftrightarrow$ $rename(\hat{r}^{-1} \circ r)(\beta) \parallel rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p))) \in G$
since $\hat{r}^{-1}$ is an extension of r^{-1}

$\Leftrightarrow$ $rename(id_{dom(r)})(\beta) \parallel rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p))) \in G$
by A27

$\Leftrightarrow$ $\beta \parallel rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p))) \in G$ and $\alpha(\beta) \subseteq dom(r)$
by the characterization of "$\preceq$" in terms of G

$\Leftrightarrow$ $\beta \preceq mirror(rename(\hat{r}^{-1})(q \parallel proj(B)(mirror(p))))$ and $\alpha(\beta) \subseteq dom(r)$

□

In the following example we show how to use the local specification synthesis technique in the IO Agent Algebra.

Example 9.5 ((Locked) IO Agent Algebra). Consider the IO agent algebra described in example 8.24. Figure 3 shows an intuitive graphical representation of the system

$$proj(\{a, b, c, d, e, f\})(\beta \parallel p_1 \parallel p_2),$$

where

$$\begin{aligned} p_1 &= (\{a, g, h\}, \{d\}), \\ p_2 &= (\{d\}, \{g, j\}) \end{aligned}$$

and β is an agent variable. Suppose we would like to solve the system for β so that it satisfies the specification

$$p' = (\{a, b\}, \{c, d\}).$$

As discussed in example 8.24, this algebra does not have mirrors, and therefore we are unable to apply our solution to the local specification synthesis. However we can embed the model in the Locked IO agent algebra described in example 8.25 as follows:

$$\begin{aligned} p_1 &\rightarrow (\{a, g, h\}, \{d\}, \emptyset), \\ p_2 &\rightarrow (\{d\}, \{g, j\}, \emptyset) \\ p' &\rightarrow (\{a, b\}, \{c, d\}, \emptyset). \end{aligned}$$

Because of the embedding, the system expression is unchanged. Thus, applying theorem 9.4 we obtain

$$proj(\{a, b, c, d, e, f\})(\beta \parallel p_1 \parallel p_2) \preceq p'$$

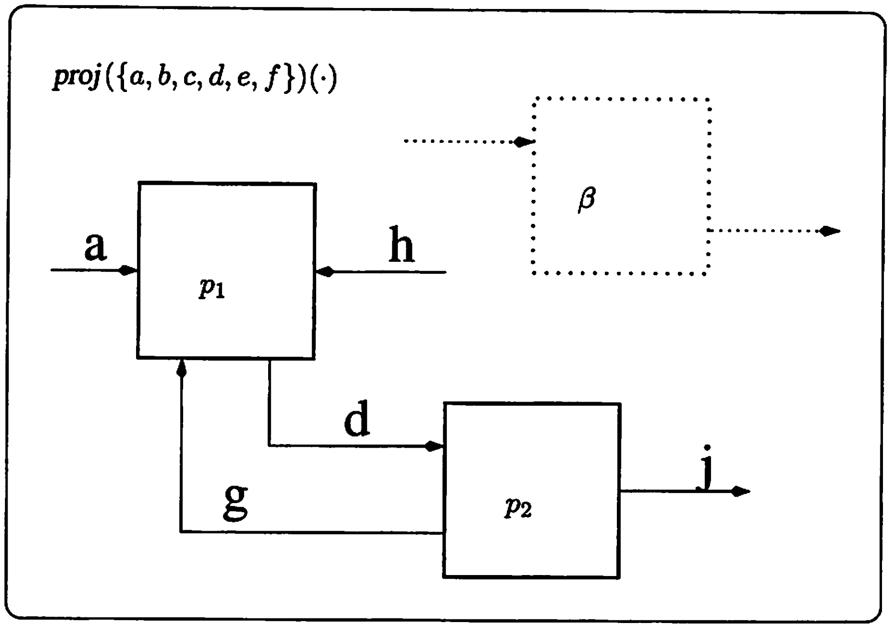

Figure 3: IO agent system

if and only if

$$\beta \preceq mirror(p_1 \parallel p_2 \parallel proj(\{a,b,c,d,e,f\})(mirror(p')))$$

Substituting the real quantities for the symbols:

$$\begin{aligned}
& mirror(p_1 \parallel p_2 \parallel proj(\{a,b,c,d,e,f\})(p')) = \\
&= mirror((\{a,g,h\},\{d\},\emptyset) \parallel (\{d\},\{g,j\},\emptyset) \parallel \\
&\quad \parallel proj(\{a,b,c,d,e,f\})(mirror((\{a,b\},\{c,d\},\emptyset)))) \\
&= mirror((\{a,h\},\{d,g,j\},\emptyset) \parallel proj(\{a,b,c,d,e,f\})(\{c,d\},\{a,b\},\mathcal{A}-\{a,b,c,d\})) \\
&= mirror((\{a,h\},\{d,g,j\},\emptyset) \parallel proj(\{a,b,c,d,e,f\})(\{c,d\},\{a,b\},\{e,f\})) \\
&= mirror((\{h,c\},\{a,b,d,g,j\},\{e,f\})) \\
&= (\{a,b,d,g,j\},\{h,c\},\mathcal{A}-\{a,b,c,d,e,f,g,h,j\})
\end{aligned}$$

Recall that $p \preceq p'$ if and only if $I \subseteq I'$, $O' \subseteq O \subseteq O' \cup L'$ and $L \subseteq L'$. If we only consider agents that have $L = \emptyset$, the agents $p = (I, O, L)$ that can be assigned to β must be such that

$$\begin{aligned}
\emptyset & \subseteq I \subseteq \{a,b,d,g,j\} \\
\{c,h\} & \subseteq O \subseteq \mathcal{A}-\{a,b,d,e,f,g,j\}
\end{aligned}$$

We interpret this result as follows. Agent p can have as input any of the inputs allowed by the specification (i.e. a and b) and any of the outputs that are already present in the system (d, g and j), whether they are retained (d) or not (g and j). It cannot have any additional input, since they would be left "unconnected" and hidden, a situation that is not allowed by the definition of the algebra. Note that p is not *required* to have any input, even though b (which is in the specification) is not already present.

That is because the order only requires that the set of inputs of the implementation be *contained* in the set of inputs of the specification.

On the other hand, p *must* have outputs c and h in order for the system to satisfy the specification. In fact, c is required by the specification and is not already present in the rest of the system, while h is an input to p_1, and it must be converted to an output in order to project it away. Agent p can also have additional outputs, but not a and b which are inputs to the system (having them as outputs would make them outputs, contrary to the specification), d, g and j, which are already outputs in the system (and thus would collide and make the parallel composition undefined), and e and f, which are retained in the projection but are not allowed by the specification.

10 Current and Future Work

These notes present the framework of agent algebra, and provide some simple examples of its use in the context of models of concurrent systems. In our current work we are concentrating on agent algebras that include models of behavior. In particular we are considering trace structure algebra [2, 4] as a general agent algebra model of concurrent behavior, and are studying different forms of refinement relationships and their characterization in terms of conformance and mirror function. Using trace strcture algebras, we are implementing the local specification synthesis technique to solve the problem of protocol conversion.

At the same time we have developed the theory of conservative approximations [6] for agent algebras (not presented in these notes) and studied conditions for the existence of an embedding from abstract to refined models. The embeddings are keys to the correct treatment of the problem of interaction of different models of computation [5, 6].

References

[1] A. Aziz, F. Balarin, R. K. Brayton, M. D. DiBenedetto, A. Saldanha, and A. L. Sangiovanni-Vincentelli. Supervisory control of finite state machines. In P. Wolper, editor, *Proceedings of Computer Aided Verification: 7th International Conference, CAV'95, Liege, Belgium, July 1995*. Springer, 1995. LNCS vol. 939.

[2] J. R. Burch. *Trace Algebra for Automatic Verification of Real-Time Concurrent Systems*. PhD thesis, School of Computer Science, Carnegie Mellon University, Aug. 1992.

[3] J. R. Burch, D. L. Dill, E. S. Wolf, and G. D. Micheli. Modeling hierarchical combinational circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'93)*, pages 612–617, November 1993.

[4] J. R. Burch, R. Passerone, and A. Sangiovanni-Vincentelli. Overcoming heterophobia: Modeling concurrency in heterogeneous systems. In M. Koutny and A. Yakovlev, editors, *Application of Concurrency to System Design*, 2001.

[5] J. R. Burch, R. Passerone, and A. Sangiovanni-Vincentelli. Using multiple levels of abstractions in embedded software design. In Henzinger and Kirsch [10], pages 324–343.

[6] J. R. Burch, R. Passerone, and A. L. Sangiovanni-Vincentelli. Modeling techniques in design-by-refinement methodologies. In *Proceedings of the Sixth Biennial World Conference on Integrated Design and Process Technology*, June 23-28 2002.

[7] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In Henzinger and Kirsch [10], pages 148–165.

[8] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1988. Also appeared as [9].

[9] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. ACM Distinguished Dissertations. MIT Press, 1989.

[10] T. A. Henzinger and C. M. Kirsch, editors. *Embedded Software*, volume 2211 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[11] R. Passerone, L. de Alfaro, T. A. Henzinger, and A. L. Sangiovanni-Vincentelli. Convertibility verification and converter synthesis: Two faces of the same coin. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'02)*, November 2002.

[12] J. C. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998.