Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents
Michell Guzmán, Sophia Knight, Santiago Quintero, Sergio Ramírez, Camilo Rueda, Frank Valencia

To cite this version:
Michell Guzmán, Sophia Knight, Santiago Quintero, Sergio Ramírez, Camilo Rueda, et al.. Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents. [Research Report] LIX, Ecole polytechnique. 2019. hal-02172415

HAL Id: hal-02172415
https://hal.archives-ouvertes.fr/hal-02172415
Submitted on 3 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract

Spatial constraint systems (scs) are semantic structures for reasoning about spatial and epistemic information in concurrent systems. We develop the theory of scs to reason about the distributed information of potentially infinite groups. We characterize the notion of distributed information of a group of agents as the infimum of the set of join-preserving functions that represent the spaces of the agents in the group. We provide an alternative characterization of this notion as the greatest family of join-preserving functions that satisfy certain basic properties. We show compositionality results for these characterizations and conditions under which information that can be obtained by an infinite group can also be obtained by a finite group. Finally, we provide algorithms that compute the distributive group information of finite groups.

1 Introduction

In current distributed systems such as social networks, actors behave more as members of a certain group than as isolated individuals. Information, opinions, and beliefs of a particular actor are frequently the result of an evolving process of interchanges with other actors in a group. This suggests a reified notion of group as a single actor operating within the context of the collective information of its members. It also conveys two notions of information, one spatial and the other epistemic. In the former, information is localized in compartments associated with a user or group. In the latter, it refers to something known or believed by a single agent or collectively by a group.
In this paper we pursue the development of a principled account of a reified notion of
group by taking inspiration from the epistemic notion of distributed knowledge [12]. A group
has its information distributed among its member agents. We thus develop a theory about
what exactly is the information available to agents as a group when considering all that is
distributed amongst its members.

In our account a group acts itself as an agent carrying the collective information of its
members. We can interrogate, for instance, whether there is a potential contradiction or
unwanted distributed information that a group might be involved in among its members
or by integrating a certain agent. This is a fundamental question since it may predict or
prevent potentially dangerous evolutions of the system.

Furthermore, in many real life multi-agent systems, the agents are unknown in advance.
New agents can subscribe to the system in unpredictable ways. Thus, there is usually no
a-priori bound on the number of agents in the system. It is then often convenient to model
the group of agents as an infinite set. In fact, in models from economics and epistemic
logic [14, 13], groups of agents have been represented as infinite, even uncountable, sets. In
accordance with this fact, in this paper we consider that groups of agents can also be infinite.
This raises interesting issues about the distributed information of such groups. In particular,
that of group compactness: information that when obtained by an infinite group can also be
obtained by one of its finite subgroups. We will provide conditions for this to hold.

Context. Constraint systems (cs)¹ are algebraic structures for the semantics of process
calculi from concurrent constraint programming (ccp) [18]. In this paper we shall study cs
as semantic structures for distributed information of a group of agents.

A cs can be formalized as a complete lattice (Con, ⊑). The elements of Con represent
partial information and we shall think of them as being assertions. They are traditionally
referred to as constraints since they naturally express partial information (e.g., \( x > 42 \)). The
order \( ⊑ \) corresponds to entailment between constraints, \( c ⊑ d \), often written \( d ⊒ c \), means \( c \)
can be derived from \( d \), or that \( d \) represents as much information as \( c \). The join \( ⊔ \), the bottom
true and the top false of the lattice correspond to conjunction, the empty information and
the join of all (possibly inconsistent) information.

The notion of computational space and the epistemic notion of belief in the spatial ccp
(sccp) process calculus [15] is represented as a family of join-preserving maps \( s_i : Con → Con \)
called space functions. A cs equipped with space functions is called a spatial constraint
system (scs). From a computational point of view \( s_i(c) \) can be interpreted as an assertion
specifying that \( c \) resides within the space of agent \( i \). From an epistemic point of view, \( s_i(c) \)
specifies that \( i \) considers \( c \) to be true. An alternative epistemic view is that \( i \) interprets \( c \) as
\( s_i(c) \). All these interpretations convey the idea of \( c \) being local or subjective to agent \( i \).

This work. In the spatial ccp process calculus sccp [15], scs are used to specify the
spatial distribution of information in configurations \( ⟨ P,c ⟩ \) where \( P \) is a process and \( c \) is a
constraint, called the store, representing the current partial information. E.g., a reduction
\( ⟨ P, s_1(a) ⊔ s_2(b) ⟩ \rightarrow ⟨ Q, s_1(a) ⊔ s_2(b ⊔ c) ⟩ \) means that \( P \), with \( a \) in the space of agent \( 1 \)
and \( b \) in the space of agent \( 2 \), can evolve to \( Q \) while adding \( c \) to the space of agent \( 2 \).

Given the above reduction, assume that \( d \) is some piece of information resulting from the
combination (join) of the three constraints above, i.e., \( d = a ⊔ b ⊔ c \), but strictly above the join
of any two of them. We are then in the situation where neither agent has \( d \) in their spaces,
but as a group they could potentially have \( d \) by combining their information. Intuitively, \( d \) is
distributed in the spaces of the group \( I = \{1,2\} \). Being able to predict the information that

¹ For simplicity we use cs for both constraint system and its plural form.
agents 1 and 2 may derive as group is a relevant issue in multi-agent concurrent systems, particularly if \( d \) represents unwanted or conflicting information (e.g., \( d = \text{false} \)).

In this work we introduce the theory of group space functions \( \Delta_I : \text{Con} \to \text{Con} \) to reason about information distributed among the members of a potentially infinite group \( I \). We shall refer to \( \Delta_I \) as the \textit{distributed space} of group \( I \). In our theory \( c \sqsupseteq \Delta_I(e) \) holds exactly when we can derive from \( c \) that \( e \) is distributed among the agents in \( I \). E.g., for \( d \) above, we should have \( s_1(a) \sqcup s_2(b \sqcup e) \sqsupseteq \Delta_{\{1,2\}}(d) \) meaning that from the information \( s_1(a) \sqcup s_2(b \sqcup e) \) we can derive that \( d \) is distributed among the group \( I = \{1,2\} \). Furthermore, \( \Delta_I(e) \sqsupseteq \Delta_J(e) \) holds whenever \( I \subseteq J \) since if \( e \) is distributed among a group \( I \), it should also be distributed in a group that includes the agents of \( I \).

Distributed information of infinite sets can be used to reason about multi-agent computations with unboundedly many agents. For example, a \textit{computation} in sccp is a possibly infinite reduction sequence \( \gamma \) of the form \( \langle P_0, c_0 \rangle \rightarrow \langle P_1, c_1 \rangle \rightarrow \cdots \) with \( c_0 \sqsubseteq c_1 \sqsubseteq \cdots \). The \textit{result} of \( \gamma \) is \( \bigsqcup_{n \geq 0} c_n \), the join of all the stores in the computation. In sccp all fair computations from a configuration have the same result [15]. Thus, the \textit{observable behaviour} of \( P \) with initial store \( c \), written \( \mathcal{O}(P,c) \), is defined as the result of any fair computation starting from \( \langle P,c \rangle \). Now consider a setting where in addition to their sccp capabilities in [15], processes can also create new agents. Hence, unboundedly many agents, say agents \( 1, 2, \ldots \), may be created during an infinite computation. In this case, \( \mathcal{O}(P,c) \sqsupseteq \Delta_N(\text{false}) \), where \( N \) is the set of natural numbers, would imply that some (finite or infinite) set of agents in any fair computation from \( \langle P,c \rangle \) may reach contradictory local information among them. Notice that from the above-mentioned properties of distributed spaces, the existence of a finite set of agents \( H \subseteq N \) such that \( \mathcal{O}(P,c) \sqsupseteq \Delta_H(\text{false}) \) implies \( \mathcal{O}(P,c) \sqsupseteq \Delta_N(\text{false}) \). The converse of this implication will be called \textit{group compactness} and we will provide meaningful sufficient conditions for it to hold.

Our main contributions are listed below.

1. We characterize the distributed space \( \Delta_I \) as a space function resulting from the infimum of the set of join-preserving functions that represent the spaces of the agents of a \textit{possibly infinite} group \( I \).
2. We provide an alternative characterization of a distributed space as the greatest join preserving function that satisfies certain basic properties.
3. We show that distributed spaces have an inherent \textit{compositional} nature: The information of a group is determined by that of its subgroups.
4. We provide a \textit{group compactness} result for groups: Given an infinite group \( I \), meaningful conditions under which \( c \sqsupseteq \Delta_I(e) \) implies \( c \sqsupseteq \Delta_J(e) \) for some finite group \( J \subseteq I \).
5. For finite scs we shall provide \textit{algorithms} to compute \( \Delta_I \) that exploit the above-mentioned compositional nature of distributed spaces.

All in all, in this paper we put forward an algebraic theory for group reasoning in the context of ccp. The theory and algorithms here developed can be used in the semantics of the spatial ccp process calculus to reason about or prevent potential unwanted evolutions of ccp processes. One could imagine the incorporation of group reasoning in a variety of process algebraic settings and indeed we expect that such formalisms will appear in due course.

## 2 Background

We presuppose basic knowledge of domain and order theory [3, 1, 6] and use the following notions. Let \( \mathcal{C} \) be a poset (\( \text{Con}, \sqsubseteq \)), and let \( S \subseteq \text{Con} \). We use \( \bigsqcup S \) to denote the least upper bound (or \textit{supremum} or \textit{join}) of the elements in \( S \), and \( \bigcap S \) is the greatest lower
Reasoning about Distributed Knowledge

4 Reasoning about Distributed Knowledge

bound (glb) (infimum or meet) of the elements in $S$. An element $e \in S$ is the greatest
element of $S$ iff for every element $e' \in S$, $e' \subseteq e$. If such $e$ exists, we denote it by $\max S$.
As usual, if $S = \{c, d\}$, $c \sqcup d$ and $c \sqcap d$ represent $\bigvee S$ and $\bigwedge S$, respectively. If $S = \emptyset$, we
denote $\bigvee S = \text{true}$ and $\bigwedge S = \text{false}$. We say that $C$ is a complete lattice iff each subset
of $\text{Con}$ has a supremum in $\text{Con}$. The poset $C$ is distributive iff for every $a, b, c \in \text{Con}$,
$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$. A non-empty set $S \subseteq \text{Con}$ is directed iff for every pair of
elements $x, y \in S$, there exists $z \in S$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$, or iff every finite subset
of $S$ has an upper bound in $S$. Also $c \in \text{Con}$ is compact iff for any directed subset $D$ of
$\text{Con}$, $c \sqsubseteq \bigvee D$ implies $c \sqsubseteq d$ for some $d \in D$. A self-map on $\text{Con}$ is a function $f$ from $\text{Con}$ to
$\text{Con}$. Let $(\text{Con}, \sqsubseteq)$ be a complete lattice. The self-map $f$ on $\text{Con}$ preserves the join of a set
$S \subseteq \text{Con}$ iff $f(\bigvee S) = \bigvee \{f(c) \mid c \in S\}$. A self-map that preserves the join of finite sets is
called join-homomorphism. A self-map $f$ on $\text{Con}$ is monotonic if $a \sqsubseteq b$ implies $f(a) \sqsubseteq f(b)$.
We say that $f$ distributes over joins (or that $f$ preserves joins) iff it preserves the join of
arbitrary sets. A self-map $f$ on $\text{Con}$ is continuous iff it preserves the join of any directed set.

3 Spatial Constraint Systems

Constraint systems [18] are semantic structures to specify partial information. They can be
formalized as complete lattices [2].

Definition 1 (Constraint Systems [2]). A constraint system (cs) $C$ is a complete lattice
$(\text{Con}, \sqsubseteq)$. The elements of $\text{Con}$ are called constraints. The symbols $\sqcap$, true and false will be
used to denote the least upper bound (lab) operation, the bottom, and the top element of $C$.

The elements of the lattice, the constraints, represent (partial) information. A constraint
can be viewed as an assertion. The lattice order $\sqsubseteq$ is meant to capture entailment of
information: $c \sqsubseteq d$, alternatively written $d \sqsupseteq c$, means that the assertion $d$ represents at
least as much information as $c$. We think of $d \supseteq c$ as saying that $d$ entails $c$ or that $c$ can
be derived from $d$. The operator $\sqcup$ represents join of information; $c \sqcup d$ can be seen as an
assertion stating that both $c$ and $d$ hold. We can think of $\sqcup$ as representing conjunction
of assertions. The top element represents the join of all, possibly inconsistent, information,
hence it is referred to as false. The bottom element true represents empty information. We
say that $c$ is consistent if $c \neq false$, otherwise we say that $c$ is inconsistent. Similarly, we say
that $c$ is consistent/inconsistent with $d$ if $c \sqcup d$ is consistent/inconsistent.

Constraint Frames. One can define a general form of implication by adapting the
corresponding notion from Heyting Algebras to cs. A Heyting implication $c \rightarrow d$ in our
setting corresponds to the weakest constraint one needs to join $c$ with to derive $d$.

Definition 2 (Constraint Frames [7]). A constraint system $(\text{Con}, \sqsubseteq)$ is said to be a constraint
frame iff its joins distribute over arbitrary meets. More precisely, $c \sqcup \bigwedge S = \bigwedge \{c \sqcup e \mid e \in S\}$
for every $c \in \text{Con}$ and $S \subseteq \text{Con}$. Define $c \rightarrow d$ as $\bigwedge \{e \in \text{Con} \mid c \sqcup e \sqsubseteq d\}$.

The following properties of Heyting implication correspond to standard logical properties
(with $\rightarrow$, $\sqcup$, and $\sqsubseteq$ interpreted as implication, conjunction, and entailment).

Proposition 3 ([7]). Let $(\text{Con}, \sqsubseteq)$ be a constraint frame. For every $c, d, e \in \text{Con}$ the
following holds: (1) $c \sqcup (c \rightarrow d) = c \sqcup d$, (2) $(c \rightarrow d) \sqsubseteq d$, (3) $c \rightarrow d = \text{true}$ iff $c \sqsubseteq d$.

Spatial Constraint Systems. The authors of [15] extended the notion of cs to account for
distributed and multi-agent scenarios with a finite number of agents, each having their own
space for local information and their computations. The extended structures are called
spatial cs (scs). Here we adapt scs to reason about possibly infinite groups of agents.
A group $G$ is a set of agents. Each $i \in G$ has a space function $s_i : \text{Con} \rightarrow \text{Con}$ satisfying some structural conditions. Recall that constraints can be viewed as assertions. Thus given $c \in \text{Con}$, we can then think of the constraint $s_i(c)$ as an assertion stating that $c$ is a piece of information residing at a space of agent $i$. Some alternative epistemic interpretations of $s_i(c)$ is that it is an assertion stating that agent $i$ believes $c$, that $c$ holds within the space of agent $i$, or that agent $i$ interprets $c$ as $s_i(c)$. All these interpretations convey the idea that $c$ is local or subjective to agent $i$.

In [15] cs are used to specify the spatial distribution of information in configurations $(\langle P, c \rangle)$ where $P$ is a process and $c$ is a constraint. E.g., a reduction $(\langle P, s_i(c) \sqcup s_j(d) \rangle) \rightarrow (\langle Q, s_i(c) \sqcup s_j(d \sqcup e) \rangle)$ means that $P$ with $c$ in the space of agent $i$ and $d$ in the space of agent $j$ can evolve to $Q$ while adding $e$ to the space of agent $j$.

We now introduce the notion of space function.

**Definition 4 (Space Functions).** A space function over a cs $(\text{Con}, \sqsubseteq)$ is a continuous self-map $f : \text{Con} \rightarrow \text{Con}$ s.t. for every $c, d \in \text{Con}$ $f(\text{true}) = \text{true}$, $(\text{S.2}) f(c \sqcup d) = f(c) \sqcup f(d)$. We shall use $\mathcal{S}(\text{C})$ to denote the set of all space functions over $\text{C} = (\text{Con}, \sqsubseteq)$.

The assertion $f(c)$ can be viewed as saying that $c$ is in the space represented by $f$. Property S.1 states that having an empty local space amounts to nothing. Property S.2 allows us to join and distribute the information in the space represented by $f$.

In [15] space functions were not required to be continuous. Nevertheless, we will argue later, in Remark 18, that continuity comes naturally in the intended phenomena we wish to capture: modelling information of possibly infinite groups. In fact, in [15] cs could only have finitely many agents.

In this work we also extend cs to allow arbitrary, possibly infinite, sets of agents. The continuity requirement in addition to S.1 and S.2 makes space functions to preserve arbitrary joins.

**Proposition 5 ([17]).** Let $f$ be a space function over a cs $(\text{Con}, \sqsubseteq)$. Then (1) $f$ is monotonic and (2) $f$ preserves arbitrary joins.

A spatial cs is a cs with a possibly infinite group of agents each having a space function.

**Definition 6 (Spatial Constraint Systems).** A spatial cs (scs) is a cs $\text{C} = (\text{Con}, \sqsubseteq)$ equipped with a possibly infinite tuple $\mathcal{S} = (s_i)_{i \in G}$ of space functions from $\mathcal{S}(\text{C})$.

We shall use $(\text{Con}, \sqsubseteq, (s_i)_{i \in G})$ to denote an scs with a tuple $(s_i)_{i \in G}$. We refer to $G$ and $\mathcal{S}$ as the group of agents and space tuple of $\text{C}$ and to each $s_i$ as the space function in $\text{C}$ of agent $i$. Subsets of $G$ are also referred to as groups of agents (or sub-groups of $G$).

Let us illustrate a simple scs.

**Example 7.** The scs $(\text{Con}, \sqsubseteq, (s_i)_{i \in \{1,2\}})$ in Fig.1 is given by the complete lattice $\mathcal{M}_2$ and two agents. We have $\text{Con} = \{p \lor \neg p, p, \neg p, p \land \neg p\}$ and $c \sqsubseteq d$ iff $c$ is a logical consequence of $d$. The top element false is $p \land \neg p$, the bottom element true is $p \lor \neg p$, and $p$ and $\neg p$ are incomparable with each other. The set of agents is $\{1, 2\}$ with space functions $s_1$ and $s_2$: For agent 1, $s_1(p) = \neg p$, $s_1(\neg p) = p$, $s_1(\text{false}) = \text{false}$, $s_1(\text{true}) = \text{true}$, and for agent 2, $s_2(p) = \text{false} = s_2(\text{false})$, $s_2(\neg p) = \neg p$, $s_2(\text{true}) = \text{true}$. The intuition is that the agent 2 sees no difference between $p$ and false while agent 1 interprets $\neg p$ as $p$ and vice versa.

More involved examples of scs include meaningful families of structures from logic and economics such as modal algebras with continuous modal operators, Kripke structures and Aumann structures (see [15]). In Ex.19 we describe the Aumann structure example from [15]. We shall also illustrate scs with infinite groups in the next section.
In this section we characterize the notion of collective information of a group of agents. We begin with some intuition. Roughly speaking, the distributed (or collective) information of a group $I$ is the join of each piece of information that resides in the space of some $i \in I$. The distributed information of $I$ w.r.t. $c$ is the distributive information of $I$ that can be derived from $c$. We are interested in formalizing whether a given $e$ can be derived from the collective information of the group $I$ w.r.t. $c$.

The following examples, which we will use throughout the paper, illustrate the above intuition.

Example 8. Consider a scs $(Con, \sqsubseteq, (s_i)_{i \in G})$ where $G = \mathbb{N}$ and $(Con, \sqsubseteq)$ is a constraint frame. Let $c \overset{\text{def}}{=} s_1(a) \sqcup s_2(a \rightarrow b) \sqcup s_3(b \rightarrow e)$. The spatial constraint $c$ specifies the situation where $a, a \rightarrow b$ and $b \rightarrow e$ are in the spaces of agent 1, 2 and 3, respectively. Neither agent holds $e$ in their space in $c$. Nevertheless, the information $e$ can be derived from the collective information of the three agents w.r.t. $c$, since from Prop.3 we have $a \sqcup (a \rightarrow b) \sqcup (b \rightarrow e) \supseteq e$.

Let us now consider an example with infinitely many agents. Let $c' \overset{\text{def}}{=} \bigcup_{i \in \mathbb{N}} s_i(a_i)$ for some increasing chain $a_0 \subseteq a_1 \subseteq \ldots$. Take $e'$ s.t. $e' \subseteq \bigcup_{i \in \mathbb{N}} a_i$. Notice that unless $e'$ is compact (see Section 2), it may be the case that no agent $i \in \mathbb{N}$ holds $e'$ in their space; e.g., if $e' \supseteq a_i$ for any $i \in \mathbb{N}$. Yet, from our assumption, $e'$ can be derived from the collective information w.r.t. $c'$ of all the agents in $\mathbb{N}$, i.e., $\bigcup_{i \in \mathbb{N}} a_i$.

The above example may suggest that the distributed information can be obtained by joining individual local information derived from $c$. Individual information of an agent $i$ can be characterized as the $i$-projection of $c$ defined thus:

Definition 9 (Agent and Join Projections). Let $C = (Con, \sqsubseteq, (s_i)_{i \in G})$ be a scs. Given $i \in G$, the $i$-agent projection of $c \in Con$ is defined as $\pi_i(c) \overset{\text{def}}{=} \bigcup \{e \mid c \supseteq s_i(e)\}$. We say that $e$ is $i$-agent derivable from $c$ iff $\pi_i(c) \supseteq e$. Given $I \subseteq G$ the $I$-join projection of a group $I$ of $c$ is defined as $\pi_I(c) \overset{\text{def}}{=} \bigcup \{\pi_i(c) \mid i \in I\}$. We say that $e$ is $I$-join derivable from $c$ iff $\pi_I(c) \supseteq e$.

The $i$-projection of an agent $i$ of $c$ naturally represents the join of all the information of agent $i$ in $c$. The $I$-join projection of group $I$ joins individual $i$-projections of $c$ for $i \in I$. This projection can be used as a sound mechanism for reasoning about distributed-information:

If $e$ is $I$-join derivable from $c$ then it follows from the distributed-information of $I$ w.r.t. $c$.

Consider the following example.
This can be interpreted as saying that the space represented by
4.2 Distributed Spaces as Maximum Spaces.
Let us consider the lattice of space functions $C_\subseteq = (\mathcal{S}(C), \subseteq)$. Suppose that $f$ and $g$ are
space functions in $C_\subseteq$ with $f \subseteq g$. Intuitively, every piece of information $c$ in the space
represented by $g$ is also in the space represented by $f$ since $f(c) \subseteq g(c)$ for every $c \in \text{Con}$. This can be interpreted as saying that the space represented by $g$ is included in the space
represented by $f$; in other words the bigger the space, the smaller the function that represents
it in the lattice $C_\subseteq$.
Following the above intuition, the order relation $\subseteq$ of $C_\subseteq$ represents (reverse) space
inclusion and the join and meet operations in $C_\subseteq$ represent intersection and union of spaces.
The biggest and the smallest spaces are represented by the bottom and the top elements of
the lattice $C_s$, here called $\lambda_\perp$ and $\lambda_\top$ and defined as follows.

**Definition 14** (Top and Bottom Spaces). For every $c \in \textrm{Con}$, define $\lambda_\perp(c) \overset{\text{def}}{=} \text{true}$,
$\lambda_\top(c) \overset{\text{def}}{=} \text{true}$ if $c = \text{true}$ and $\lambda_\top(c) \overset{\text{def}}{=} \text{false}$ if $c \neq \text{true}$.

The distributed space $\Delta_I$ of a group $I$ can be viewed as the function that represents the
smallest space that includes all the local information of the agents in $I$. From the above
intuition, $\Delta_I$ should be the greatest space function below the space functions of the agents in
$I$. The existence of such a function follows from completeness of $(S(C), \subseteq_s)$ (Lemma 13).

**Definition 15** (Distributed Space Functions). Let $C$ be a scs $(\subseteq, (s_i)_{i \in G})$. The dis-
tributed spaces of $C$ is given by $\Delta = (\Delta_I)_{I \subseteq G}$ where

$$\Delta_I \overset{\text{def}}{=} \max \{ f \in S(C) \mid f \subseteq s_i \text{ for every } i \in I \}.$$  

We shall say that $c$ is distributed among $I \subseteq G$ w.r.t. $c$ if $c \supseteq \Delta_I(c)$. We shall refer to each
$\Delta_I$ as the (distributed) space of the group $I$.

It follows from Lemma 13 that $\Delta_I = \bigcap \{ s_i \mid i \in I \}$ (where $\bigcap$ is the meet in the complete
lattice $(S(C), \subseteq_s)$). Fig.2b illustrates a scs and its distributed space $\Delta_{\{1,2\}}$.

Compositionality. Distributed spaces have pleasant compositional properties. They capture the intuition that the distributed information of a group $I$ can be obtained from the distributive information of its subgroups.

**Theorem 16.** Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a scs $(\subseteq, (s_i)_{i \in G})$. Suppose that $K, J \subseteq I \subseteq G$. (1) $\Delta_I = \lambda_\top$ if $I = \emptyset$, (2) $\Delta_I = s_i$ if $I = \{i\}$, (3) $\Delta_J(a) \sqcup \Delta_K(b) \supseteq \Delta_I(a \sqcup b)$, and (4) $\Delta_J(a) \sqcup \Delta_K(a \to c) \supseteq \Delta_I(c)$ if $(\subseteq, \text{Con})$ is a constraint frame.

Recall that $\lambda_\top$ corresponds to the empty space (see Def.14). The first property realizes the
intuition that the empty subgroup $\emptyset$ does not have any information whatsoever distributed
w.r.t. a consistent $c$: for if $c \supseteq \Delta_\emptyset(e)$ and $e \neq \text{false}$ then $e = \text{true}$. Intuitively, the second
property says that the function $\Delta_I$ for the group of one agent must be the agent’s space
function. The third property states that a group can join the information of its subgroups.

The last property uses constraint implication, hence the constraint frame condition, to express
that by joining the information $a$ and $a \to c$ of their subgroups, the group $I$ can obtain $c$.

Let us illustrate how to derive information of a group from smaller ones using Thm.16.

**Example 17.** Let $c = s_1(a) \sqcup s_2(a \to b) \sqcup s_3(b \to c)$ as in Ex.8. We want to prove that $c$
is distributed among $I = \{1,2,3\}$ w.r.t. $c$, i.e., $c \supseteq \Delta_{\{1,2,3\}}(c)$. Using Properties 2 and 4
in Thm.16 we obtain $c \supseteq s_1(a) \sqcup s_2(a \to b) = \Delta_{\{1\}}(a) \sqcup \Delta_{\{2\}}(a \to b) \supseteq \Delta_{\{1,2\}}(b)$, and then
$c \supseteq \Delta_{\{1,2\}}(b) \sqcup s_3(b \to c) = \Delta_{\{1,2\}}(b) \sqcup \Delta_{\{3\}}(b \to c) \supseteq \Delta_{\{1,2,3\}}(c)$ as wanted.

**Remark 18** (Continuity). The example with infinitely many agents in Ex.8 illustrates well
why we require our spaces to be continuous in the presence of possibly infinite groups. Clearly $c' = \bigsqcup_{i \in N} s_i(a_i) \supseteq \bigsqcup_{i \in N} \Delta_N(a_i)$. By continuity, $\bigsqcup_{i \in N} \Delta_N(a_i) = \Delta_N(\bigsqcup_{i \in N} a_i)$ which indeed
 captures the idea that each $a_i$ is in the distributed space $\Delta_N$.

In Thm.16 we listed some useful properties about $(\Delta_I)_{I \subseteq G}$. In the next section we shall see that $(\Delta_I)_{I \subseteq G}$ is the greatest solution of three basic properties.

We conclude this subsection with an important family of scs’s from mathematical eco-
nomics: Aumann structures. We illustrate that the notion of distributed knowledge in these
structures is an instance of a distributed space.
We now wish to single out a few fundamental properties on tuples of self-maps that can be used to characterize distributed spaces.

**Definition 20** (Distribution Candidates). Let $\mathbf{C}$ be a scs $(\text{Con}, \sqsubseteq, (s_i)_{i \in G})$. A tuple $\delta = (\delta_I)_{I \subseteq G}$ of self-maps on Con is a group distribution candidate (gdc) of $\mathbf{C}$ if for each $I, J \subseteq G$:

1. (D.1) $\delta_I$ is a space function in $\mathbf{C}$,
2. (D.2) $\delta_I = s_i$ if $I = \{i\}$,
3. (D.3) $\delta_I \supseteq \delta_J$ if $I \subseteq J$.

Property D.1 requires each $\delta_I$ to be a space function. This is trivially met for $\delta_I = \Delta_I$. Property D.2 says that the function $\delta_I$ for a group of one agent must be the agent’s space function. Clearly, $\delta_{\{i\}} = \Delta_{\{i\}}$ satisfies D.2; indeed the distributed space of a single agent is their own space. Finally, Property D.3 states that $\delta_I(c) \supseteq \delta_J(c)$, if $I \subseteq J$. This is also trivially satisfied if we take $\delta_I = \Delta_I$ and $\delta_J = \Delta_J$. Indeed if a subgroup $I$ has some distributed information $c$ then any subgroup $J$ that includes $I$ should also have $c$. This also realizes our intuition above: The bigger the group, the bigger the space and thus the smaller the space function that represents it.

**Figure 2** Projections (a) and Distributed Space function (b) over lattice $M_2$. 

5.3 Distributed Spaces as Group Distributions Candidates.

We now wish to single out a few fundamental properties on tuples of self-maps that can be used to characterize distributed spaces.
Properties D1-D3, however, do not determine $\Delta$ uniquely. In fact, there could be infinitely-many tuples of space functions that satisfy them. For example, if we were to choose $\delta_0 = \lambda_T$, $\delta_{(i)} = s_i$ for every $i \in G$, and $\delta_I = \lambda_L$ whenever $|I| > 1$ then D1, D2 and D3 would be trivially met. But these space functions would not capture our intended meaning of distributed spaces: E.g., we would have $true \sqsupseteq \delta_I(e)$ for every $e$ thus implying that any $e$ could be distributed in the empty information $true$ amongst the agents in $I \neq \emptyset$.

Nevertheless, the following theorem states that $(\Delta_I)_{I \subseteq G}$ could have been equivalently defined as the greatest space functions satisfying Properties D1-D3.

> **Theorem 21** (Max gdc). Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. Then $(\Delta_I)_{I \subseteq G}$ is a gdc of $C$ if and only if $(\delta_I)_{I \subseteq G}$ is a gdc of $C$ then $\delta_I \sqsubseteq_\Delta \Delta_I$ for each $I \subseteq G$.

Let us illustrate the use of Properties D1-D3 in Thm.21 with the following example.

> **Example 22.** Let $c = s_1(a \sqcup b) \sqcup s_3(b \rightarrow c)$ as in Ex.8. We want to prove $c \sqsubseteq \Delta_I(e)$ for $I = \{1, 2, 3\}$. From D.2 we have $c = \Delta_{(1)}(a) \sqcup \Delta_{(2)}(a \rightarrow b) \sqcup \Delta_{(3)}(b \rightarrow e)$. We can then use D.3 to obtain $c \sqsupseteq \Delta_I(a) \sqcup \Delta_I(a \rightarrow b) \sqcup \Delta_I(b \rightarrow e)$. Finally, by D.1 and Proposition 3 we infer $c \sqsupseteq \Delta_I(a \sqcup (a \rightarrow b) \sqcup (b \rightarrow e)) \sqsupseteq \Delta_I(e)$, thus $c \sqsupseteq \Delta_I(e)$ as wanted. Now consider our counter-example in Ex.11 with $d = s_1(b) \sqcup s_2(b)$. We wish to prove $d \sqsupseteq \Delta_I(b)$ for $I = \{1, 2\}$. I.e., that $b$ can be derived from $d$ as being in a space of a member of $\{1, 2\}$. Using D.1 and D.3 we obtain $d \sqsupseteq d' = \Delta_{(1)}(b) \sqcup \Delta_{(2)}(b) \sqsupseteq \Delta_{(1, 2)}(b) \sqcap \Delta_{(2, 1)}(b) = \Delta_{(2, 1)}(b)$ as wanted.

The characterization of distributed spaces by Thm.21 provide us with a convenient proof method: E.g, to prove that a tuple $F = (f_I)_{I \subseteq G}$ equals $(\Delta_I)_{I \subseteq G}$, it suffices to show that the tuple is a gdc and that $f_I \sqsubseteq_\Delta \Delta_I$ for all $I \subseteq G$. We use this mechanism in Section 5.

### 4.4 Group Projections

As promised in Section 4.1 we now give a definition of Group Projection. The function $\Pi_I(c)$ extracts exactly all information that the group $I$ may have distributed w.r.t. $c$.

> **Definition 23** (Group Projection). Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. Given the set $I \subseteq G$, the $I$-group projection of $c \in Con$ is defined as $\Pi_I(c) \overset{\text{def}}{=} \bigcup \{e \mid c \sqsupseteq \Delta_I(e)\}$. We say that $e$ is $I$-group derivable from $c$ iff $\Pi_I(c) \sqsupseteq e$.

Much like space functions and agent projections, group projections and distributed spaces also form a pleasant correspondence: a Galois connection [3].

> **Proposition 24.** Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. For every $c, e \in Con$, (1) $c \sqsubseteq \Pi_I(c)$ iff $\Pi_I(c) \sqsupseteq e$, (2) $\Pi_I(c) \sqsubseteq \Pi_J(c)$ if $J \subseteq I$, and (3) $\Pi_I(c) \sqsubseteq \pi_I(c)$.

The first property in Prop.24, a Galois connection, states that we can conclude from $c$ that $e$ is in the distributed space of $I$ exactly when $e$ is $I$-group derivable from $c$. The second says that the bigger the group, the bigger the projection. The last property says that whatever is $I$-join derivable is $I$-group derivable, although the opposite is not true as shown in Ex.11.

### 4.5 Group Compactness.

Suppose that an infinite group of agents $I$ can derive $e$ from $c$ (i.e., $c \sqsubseteq \Delta_I(e)$). A legitimate question is whether there exists a finite sub-group $J$ of agents from $I$ that can also derive $e$ from $c$. The following theorem provides a positive answer to this question provided that $e$ is a compact element (see Section 2) and $I$-join derivable from $c$.
Theorem 25 (Group Compactness). Let \((\Delta_I)_{I \subseteq G}\) be the distributed spaces of an scs \(C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G})\). Suppose that \(c \sqsupseteq \Delta_I(c)\). If \(c\) is compact and \(I\)-join derivable from \(c\) then there exists a finite set \(J \subseteq I\) such that \(c \sqsupseteq \Delta_J(c)\).

We conclude this section with the following example of group compactness.

Example 26. Consider the example with infinitely many agents in Ex.8. We have \(c' = \bigsqcup_{i \in \mathbb{N}} s_i(a_i)\) for some increasing chain \(a_0 \sqsubseteq a_1 \sqsubseteq \ldots\) and \(c'\) s.t. \(c' \sqsubseteq \bigsqcup_{i \in \mathbb{N}} a_i\). Notice that \(c' \sqsupseteq \Delta_N(c')\) and \(\tau_N(c') \equiv c'\). Hence \(c'\) is \(\mathbb{N}\)-join derivable from \(c'\). If \(c'\) is compact, by Thm.25 there must be a finite subset \(J \subseteq \mathbb{N}\) such that \(c' \sqsupseteq \Delta_J(c')\).

5 Computing Distributed Information

Let us consider a finite scs \(C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G})\) with distributed spaces \((\Delta_I)_{I \subseteq G}\). By finite scs we mean that \(\text{Con}\) and \(G\) are finite sets. Let us consider the problem of computing \(\Delta_I\):

Given a set \(\{s_i\}_{i \in I}\) of space functions, we wish to find the greatest space function \(f\) such that \(f \sqsubseteq s_i\) for all \(i \in I\) (see Def.15).

Because of the finiteness assumption, the above problem can be rephrased in simpler terms: Given a finite lattice \(L\) and a finite set \(S\) of join-homomorphisms on \(L\), find the greatest join-homomorphism below all the elements of \(S\). Even in small lattices with four elements and two space functions, finding such greatest function may not be immediate, e.g., for \(S = \{s_1, s_2\}\) and the lattice in Fig.1 the answer is given Fig.2b.

In this section we shall use the theory developed in previous sections to help us find algorithms for this problem. Recall from Def.15 and Lemma 13 that \(\Delta_I\) equals the following

\[
\max\{f \in S(C) \mid f \sqsubseteq s_i \text{ for all } i \in I\} = \bigsqcup\{f \in S(C) \mid f \sqsubseteq s_i \text{ for all } i \in I\} = \prod\{s_i \mid i \in I\}
\]

A naive (meet-based) approach would be to compute \(\Delta_I(c)\) by taking the point-wise meet construction \(\sigma_I(c) \overset{\text{def}}{=} \prod\{s_i(c) \mid i \in I\}\) for each \(c \in \text{Con}\). But this does not work in general since \(\Delta_I(c) = \prod\{s_i \mid i \in I\}(c)\) is not necessarily equal to \(\sigma_I(c) = \prod\{s_i(c) \mid i \in I\}\). In fact \(\sigma_I \sqsupseteq \Delta_I\) but \(\sigma_I\) may not even be a space function as shown in Fig.3a.

A brute force (join-based) solution to computing \(\Delta_I(c)\) can be obtained by generating the set \(\{f(c) \mid f \in S(C)\} \sqsubseteq s_I\) for all \(i \in I\) and taking its join. This approach works since the join of a set of space functions \(S\) can be computed point-wise: \((\bigsqcup S)c = \bigsqcup\{f(c) \mid f \in S\}\).

However, the number of such functions in \(S(C)\) can be at least factorial in the size of \(\text{Con}\). For constraint frames, which under the finite assumption coincides with distributive lattices, the size of \(S(C)\) can be non-polynomial in the size of \(\text{Con}\).

Proposition 27 (Lower Bounds on Number of Space Functions). For every \(n \geq 2\), there exists a cs \(C = (\text{Con}, \sqsubseteq)\) such that \(|S(C)| \geq (n - 2)!\) and \(n = |\text{Con}|\). For every \(n \geq 1\), there exists a constraint frame \(C = (\text{Con}, \sqsubseteq)\) such that \(|S(C)| \geq n^{\log n}\) and \(n = |\text{Con}|\).

Nevertheless, in the following sections we shall be able to exploit order theoretical results and properties of distributed spaces to compute \(\Delta_I(c)\) for every \(c \in \text{Con}\) in polynomial time in the size of \(\text{Con}\). The first approach uses the inherent compositional nature of \(\Delta_I\) in distributed lattices. The second approach uses the above-mentioned \(\sigma\) as suitable upper bound of \(\Delta_I\) to compute \(\Delta_I(c)\) by approximating it from above.

5.1 Distributed Spaces in Distributed Lattices

Here we shall illustrate some pleasant compositionality properties of distributed spaces that can be used for computing \(\Delta_I\) in distributed lattices (constraint frames). These properties
(a) For \( I = \{1, 2\} \), \( \sigma_I(c) = \bigcap_{i \in I} s_i(c) \) is not a space function: \( \sigma_I(p \lor \neg p) \neq \sigma_I(p) \lor \sigma_I(\neg p) \).

(b) For \( I = \{1, 2\} \), \( \delta^+_I \) (Lemma 28) is not a space function: \( \delta^+_I(b \lor \delta^+_I(e)) = b \neq \delta^+_I(b \lor e) \).

**Figure 3** Counter-examples over lattice \( M_2 \) (a) and the non-distributive lattice \( M_3 \) (b).

capture the intuition that just like distributed information of a group \( I \) is the collective information from all its members, it is also the collective information of its subgroups. The following results can be used to produce algorithms to compute \( \Delta_I(c) \).

We use \( X^J \) to denote the set of tuples \( (x_j)_{j \in J} \) of elements \( x_j \in X \) for each \( j \in J \).

**Lemma 28.** Let \( (\Delta_I)_{I \subseteq G} \) be the distributed spaces of a finite scs \( C = (\text{Con}, \subseteq, (s_i)_{i \in G}) \). Suppose that \( (\text{Con}, \subseteq) \) is a constraint frame. Let \( \delta^+_I : \text{Con} \to \text{Con} \), with \( I \subseteq G \), be the function \( \delta^+_I(c) \overset{\text{def}}{=} \bigcap\{\bigcup_{i \in I} s_i(a_i) \mid (a_i)_{i \in I} \in \text{Con}^I \text{ and } \bigcup_{i \in I} a_i \subseteq c\} \). Then \( \Delta_I = \delta^+_I \).

The above lemma basically says that \( \Delta_I(c) \) is the greatest information below all possible combinations of information in the spaces of the agents in \( I \) that derive \( c \). The proof that \( \delta^+_I \subseteq \Delta_I \) uses the fact that space functions preserve joins. The proof that \( \delta^+_I \subseteq \Delta_I \) proceeds by showing that \( (\delta^+_I)_{I \subseteq G} \) is a group distribution candidate (Def.20). Distributivity of the lattice \( (\text{Con}, \subseteq) \) is crucial for this direction. In fact without it \( \Delta_I = \delta^+_I \) does not necessarily hold as shown by the following counter-example.

**Example 29.** Consider the non-distributive lattice \( M_3 \) and the space functions \( s_1 \) and \( s_2 \) in Figure 3b. We obtain \( \delta^+_J(b \lor c) = \delta^+_J(c) = a \) and \( \delta^+_J(b \lor \delta^+_J(c)) = b \lor a = b \). Then, \( \delta^+_J(b \lor c) \neq \delta^+_J(b \lor \delta^+_J(c)) \), i.e., \( \delta^+_J \) is not a space function.

Lemma 28 can be used to prove the following theorem which intuitively characterizes the information of a group from that of its subgroups. Each of the following results will be used to generate algorithms to compute \( \Delta_I(c) \), each an improvement on the previous one.

**Theorem 30.** Let \( (\Delta_I)_{I \subseteq G} \) be the distributed spaces of a finite scs \( C = (\text{Con}, \subseteq, (s_i)_{i \in G}) \). Suppose that \( (\text{Con}, \subseteq) \) is a constraint frame. Let \( J, K \subseteq G \) be two groups such that \( I = J \cup K \). Then the following equalities hold:

1. \( \Delta_I(c) = \bigcap \{\Delta_J(a) \cup \Delta_K(b) \mid a, b \in \text{Con} \text{ and } a \cup b \supseteq c\} \). (1)
2. \( \Delta_I(c) = \bigcap \{\Delta_J(a) \cup \Delta_K(a \rightarrow c) \mid a \in \text{Con}\} \). (2)
3. \( \Delta_I(c) = \bigcap \{\Delta_J(a) \cup \Delta_K(a \rightarrow c) \mid a \in \text{Con} \text{ and } a \supseteq c\} \). (3)
The above properties bear witness to the inherent compositional nature of our notion of distributed space. This nature will be exploited by the algorithms below. The first property in Thm.30 essentially reformulates Lemma 28 in terms of subgroups rather than agents. It can be proven by replacing $\Delta_I(a)$ and $\Delta_K(b)$ by $\delta_I^+(a)$ and $\delta_K^+(b)$, defined in Lemma 28 and using distributivity of joins over meets. The second and third properties in Theorem 30 are pleasant simplifications of the first using heyting implication. These properties realize the intuition that by joining the information $a$ and $a \to c$ of their subgroups, the group $I$ can obtain $c$.

5.2 Algorithms for Distributed Lattices

Recall that $\lambda_\top$ represents the empty distributed space (see Def.14). Given finite scs $C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G})$ with distributed spaces $(\Delta_I)_{I \subseteq G}$, the recursive function $\text{DELPART3}(I, c)$ in Algorithm 1 computes $\Delta_I(c)$ for any given $c \in \text{Con}$. Its correctness, assuming that $(\text{Con}, \sqsubseteq)$ is a constraint frame (i.e., a distributed lattice), follows from Thm.30(3). Termination follows from the finiteness of $C$ and the fact the sets $J$ and $K$ in the recursive calls form a partition of $I$. Notice that we select a partition (in halves) rather than any two sets $K, J$ satisfying the condition $J = J \cup K$ to avoid significant recalculation.

Algorithm 1 Function $\text{DELPART3}(I, c)$ computes $\Delta_I(c)$

1: function $\text{DELPART3}(I, c)$ $\triangleright$ Computes $\Delta_I(c)$
2: if $I = \emptyset$ then
3: return $\lambda_\top(c)$
4: else if $I = \{i\}$ then
5: return $s_i(c)$
6: else
7: $\{J, K\} \leftarrow \text{PARTITION}(I)$ $\triangleright$ returns a partition $\{J, K\}$ of $I$ s.t., $|J| = |I|/2$
8: return $\bigcap\{\text{DELPART3}(J, a) \sqcup \text{DELPART3}(K, a \to c) \mid a \in \text{Con} \text{ and } a \sqsubseteq c\}$.

Algorithms. Notice $\text{DELPART3}(I, c)$ computes $\Delta_I(c)$ using Thm.30(3). By modifying Line 8 with the corresponding meet operations, we obtain two variants of $\text{DELPART3}$ that use, instead of Thm.30(3), the Properties Thm.30(1) and Thm.30(2). We call them $\text{DELPART1}$ and $\text{DELPART2}$. Finally, we also obtain a non-recursive algorithm that outputs $\Delta_I(c)$ by computing $\delta_I^+(c)$ in Lemma 28 in the obvious way: Computing the meet of elements of the form $\bigcup_{i \in I} s_i(a_i)$ for every tuple $(a_i)_{i \in I}$ such that $\bigcup_{i \in I} a_i \sqsubseteq c$. We call it $\text{DELPART}+$.  

Worst-case time complexity. We assume that binary distributive lattice operations $\sqcap, \sqcup$, and $\to$ are computed in $O(1)$ time. We also assume a fixed group $I$ of size $m = |I|$ and express the time complexity for computing $\Delta_I$ in terms of $n = |\text{Con}|$, the size of the set of constraints. The above-mentioned algorithms compute the value $\Delta_I(c)$. The worst-case time complexity for computing the function $\Delta_I$ is in (1) $O(mn^{1+m})$ using $\text{DELPART}+$, (2) $O(mn^{1+2\log_2 m})$ using $\text{DELPART1}$, and (3) $O(mn^{1+\log_2 m})$ using $\text{DELPART2}$ and $\text{DELPART3}$.

5.3 Algorithm for Arbitrary Lattices

Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G})$. The maximum space function $\Delta_I$ under a collection $(s_i)_{i \in I}$ can be computed by successive approximations, starting with some (not necessarily space) function known to be less than all $(s_i)_{i \in I}$. Assume a self map $\sigma : \text{Con} \to \text{Con}$ such that $\sigma \sqsupseteq \Delta_I$ and, for all $i \in I$, $\sigma \sqsubseteq s_i$. A good starting
point is \( \sigma(u) = \prod \{ s_i(u) \mid i \in I \} \), for all \( u \in Con \). By definition of \( \cap \), \( \sigma(u) \) is the biggest function under all functions in \( \{ s_i \}_{i \in I} \), hence \( \sigma \sqsupseteq \Delta_I \). The algorithm computes decreasing upper bounds of \( \Delta_I \) by correcting \( \sigma \) values not conforming to the space function property 
\[
\sigma(u) \cup \sigma(v) = \sigma(u \sqcup v).
\]
The correction decreases \( \sigma \) and maintains the invariant \( \sigma \sqsupseteq \Delta_I \).

There are two ways of correcting \( \sigma \) values: (1) when \( \sigma(u) \cup \sigma(v) \sqsupseteq \sigma(u \sqcup v) \), assign 
\[
\sigma(u \sqcup v) \leftarrow \sigma(u) \cup \sigma(v)
\]
and also \( \sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v) \). It can be shown that the assignments in both cases should decrease \( \sigma \) while preserving the \( \sigma \sqsupseteq \Delta_I \) invariant.

The procedure (see Algorithm 2) loops through pairs \( u, v \in Con \) while there is some pair satisfying cases (1) or (2) above for the current \( \sigma \). When there is, it updates \( \sigma \) as mentioned before. At the end of the loop all \( u, v \in Con \) pairs satisfy the space function property. By the invariant mentioned above, this means \( \sigma = \Delta_I \).

<table>
<thead>
<tr>
<th>Algorithm 2</th>
<th>\textsc{DeltaGen} finds ( \Delta_I )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma(u) \leftarrow \prod { s_i(u) \mid i \in I } ) &amp; ( \triangleright ) for all ( u \in Con )</td>
<td></td>
</tr>
<tr>
<td>\textbf{while} ( u, v \in Con \land \sigma(u) \cup \sigma(v) \neq \sigma(u \sqcup v) ) \textbf{do} &amp;</td>
<td></td>
</tr>
<tr>
<td>\quad \textbf{if} ( \sigma(u) \cup \sigma(v) \sqsubset \sigma(u \sqcup v) ) \textbf{then} &amp; ( \triangleright ) case (1)</td>
<td></td>
</tr>
<tr>
<td>\qquad \sigma(u \sqcup v) \leftarrow \sigma(u) \cup \sigma(v) &amp;</td>
<td></td>
</tr>
<tr>
<td>\quad \textbf{else} &amp; ( \triangleright ) case (2)</td>
<td></td>
</tr>
<tr>
<td>\qquad \sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v) &amp;</td>
<td></td>
</tr>
<tr>
<td>\qquad \sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v) &amp;</td>
<td></td>
</tr>
</tbody>
</table>

The complexity of the initialization of \textsc{DeltaGen} is \( O(nm) \), where \( n = |Con| \) and \( m \) is the number of space functions. Each element in \( Con \) can be decreased at most \( n \) times. Identifying an element to be decreased (in the test of the loop) takes \( O(n^2) \). Since there are \( n^2 \) possible decreases, worst time complexity of the loop is in \( O(n^4) \).

### 6 Conclusions and Related Work

We developed semantic foundations and provided algorithms for reasoning about the distributed information of groups in multi-agents systems. We plan to develop similar techniques for reasoning about other group phenomena in multi-agent systems from social sciences and computer music such as group polarization [4] and group improvisation [17].

The closest related work is that of [15] (and its extended version [16]) which introduces spatial constraint systems (scs) for the semantics of a spatial ccp language. Their work is confined to a finite number of agents and to reasoning about agents individually rather than as groups. We added the continuity requirement to the space functions of [15] to be able to reason about possibly infinite groups. In [7, 8, 9, 10] scs are used to reason about beliefs, lies and other epistemic utterances but also restricted to a finite number of agents and individual, rather than group, behaviour of agents.

Our work is inspired by the epistemic concept of distributed knowledge [5]. Knowledge in distributed systems was discussed in [11], based on interpreting distributed systems using Hintikka’s notion of possible worlds. In this definition of distributed knowledge, the system designer ascribes knowledge to processors (agents) in each global state (a processor’s local state). In [12] the authors present a general framework to formalize the knowledge of a group of agents, in particular the notion of distributed knowledge. The authors consider distributed knowledge as knowledge that is distributed among the agents belonging to a given group, without any individual agent necessarily having this knowledge. In [13] the
authors study knowledge and common knowledge in situations with infinitely many agents. The authors highlight the importance of reasoning about infinitely many agents in situations where the number of agents is not known in advance. Their work does not address distributed knowledge but points out potential technical difficulties in their future work.

References

A Proofs

Proof of Lemma 13.
Let \( C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G}) \) be a spatial cs. Then \( C_s \) is a complete lattice.

**Proof.** We are going to prove that the set of space functions over \( (\text{Con}, \sqsubseteq) \) forms a complete lattice.

Let \( S = \{s_i\}_{i \in I} \) be a family of space functions with arbitrary indexing set \( I \subseteq G \). For every constraint \( c \in \text{Con} \), we let \( (\bigsqcup S)(c) = \bigsqcup_{i \in I} s_i(c) \). We shall show that \( \bigsqcup S \) is a space function.

\[
(\bigsqcup S)(true) = \bigsqcup_{i \in I} s_i(true) = \bigsqcup true = true.
\]
\[
(\bigsqcup S)(c \sqcup d) = \bigsqcup_{i \in I} s_i(c \sqcup d)
\]
\[
= \bigsqcup_{i \in I} (s_i(c) \sqcup s_i(d))
\]
\[
= (\bigsqcup_{i \in I} s_i(c)) \sqcup (\bigsqcup_{i \in I} s_i(d))
\]
\[
= (\bigsqcup S)(c) \sqcup (\bigsqcup S)(d)
\]

Continuity: \( (\bigsqcup S)(\bigsqcup D) = \bigsqcup_{d \in D} (\bigsqcup S)(d) \) for any directed set \( D \).
Suppose that \( D \) is a directed set. From definition \( (\bigsqcup S)(\bigsqcup D) = \bigsqcup_{i \in I} s_i(\bigsqcup D) \). By the continuity of each space function \( s_i \), \( \bigsqcup_{i \in I} s_i(\bigsqcup D) = \bigsqcup_{i \in I, d \in D} s_i(d) = \bigsqcup_{d \in D} \bigsqcup_{i \in I} s_i(d) = \bigsqcup_{d \in D} (\bigsqcup S)(d) \), as required.

**Proof of Theorem 16.**
Let \( (\Delta_I)_{I \subseteq G} \) be the distributed spaces of a scs \( C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G}) \). Suppose that \( I \subseteq G \).

1. \( \Delta_I = \lambda_T \) if \( I = \emptyset \),
2. \( \Delta_I = s_i \) if \( I = \{i\} \),
3. \( \Delta_J(a) \sqcup \Delta_K(b) \subseteq \Delta_I(a \sqcup b) \) if \( K, J \subseteq I \),
4. \( \Delta_J(a) \sqcup \Delta_K(a \to c) \supseteq \Delta_I(c) \) if \( K, J \subseteq I \) and \( (\text{Con}, \sqsubseteq) \) is a constraint frame.

**Proof.** The proof of part (1) follows from definition of \( \lambda_T \) (see Definition 14) and the fact that \( \max \emptyset = true \). Part (2) is immediate from Definition 15 of \( \Delta_I \). For property (3), recall that the bigger the group the smaller the space function associated to it. Thus, if \( K, J \subseteq I \) note that \( \Delta_J(a) \sqsupseteq \Delta_I(a) \) and \( \Delta_K(b) \supseteq \Delta_I(b) \), then \( \Delta_J(a) \sqcup \Delta_K(b) \supseteq \Delta_I(a \sqcup b) \).
To prove (4), we use part (3) with \( a = a \) and \( b = a \to c \), and Proposition 3.

**Proof of Theorem 21.**
Let \( (\Delta_I)_{I \subseteq G} \) be the distributed spaces of a scs \( C = (\text{Con}, \sqsubseteq, (s_i)_{i \in G}) \). Then \( (\Delta_I)_{I \subseteq G} \) is a gcd of \( C \) and (2) if \( (\delta_I)_{I \subseteq G} \) is a gcd of \( C \) then \( \delta_I \sqsubseteq_{s} \Delta_I \) for each \( I \subseteq G \).

**Proof.** To prove (1) consider the properties (D.1)-(D.3) in Definition 20 for \( \Delta_I \). Property (D.1) follows from definition of \( \Delta_I \) (Definition 15). Property (D.2) is proven in Theorem 16 part (2). Property (D.3) is a consequence of: the bigger the group the smaller the space function associated to it. To prove (2) note that \( \delta_I \) is a space function and \( \Delta_I \) is the maximum of the space functions below \( s_i \) for every \( i \in I \).
Proof of Proposition 24.

Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. (1) If $c \preceq \Delta_I(e)$ if and only if $\Pi_I(e) \sqsubseteq e$. (2) $\Pi_I \sqsubseteq \Pi_J$ if $J \subseteq I$, and (3) $\Pi_I \sqsubseteq \pi_I$.

Proof. For (1), firstly assume that $c \preceq \Delta_I(e)$. From Definition 23, $\Pi_I(e) = \bigcup \{d \mid c \preceq \Delta_I(d)\}$. Then, $\Pi_I(e) \sqsubseteq e$, secondly, assume that $\Pi_I(e) \sqsubseteq e$ and let $S = \{d \mid c \preceq \Delta_I(d)\}$. Note that $c \sqsupseteq \bigcup \{\Delta_I(d) \mid d \in S\}$. From continuity of $\Delta_I$, we know that $\bigcup \{\Delta_I(d) \mid d \in S\} = \Delta_I(S)$. Thus, $c \preceq \Delta_I(S)$. By monotonicity $c \preceq \Delta_I(\sqcup S) = \Delta_I(\Pi_I(e)) \sqsubseteq \Delta_I(e)$.

Property (2) follows from Theorem 21. If $J \subseteq I$, then $\Delta_J \sqsubseteq \Delta_I$. Then $\{d \mid c \preceq \Delta_J(d)\} \subseteq \{d \mid c \preceq \Delta_I(d)\}$ and thus $\Pi_J \sqsubseteq \Pi_I$.

Finally, to prove property (3), by part (2) it is true that for every $i \in I$, $\Pi_I \sqsubseteq \Pi_{\{i\}}$. Therefore, $\Pi_I \sqsubseteq \bigcup_{i \in I} \Pi_{\{i\}}$. Now, for every $i \in Con$, $\bigcup_{i \in I} \Pi_{\{i\}}(c) = \bigcup_{i \in I} \bigcup \{d \mid c \preceq \Delta_{\{i\}}(d)\} = \bigcup_{i \in I} \{\pi_i(c)\} = \pi_I(c)$. Therefore, $\Pi_I(e) \sqsubseteq \pi_I(c)$, for every $c \in Con$.

Proof of Theorem 25.

Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. Suppose that $c \preceq \Delta_I(e)$. If $e$ is compact and $I$-join derivable from $c$ then there exists a finite set $J \subseteq I$ such that $c \preceq \Delta_J(e)$.

Proof. Suppose that $c \preceq \Delta_I(e)$. If $I$ is finite then take $J = I$. If $I$ is not finite, since $e$ is $I$-join derivable from $c$ we have $\pi_I(c) = \bigcup S \sqsubseteq e$ where $S = \{\pi_i(c) \mid i \in I\}$.

Define $S_I = \{\pi_I(c) \mid J \subseteq I$ and $J$ is finite $\}$. Take any $\pi_H(c), \pi_K(c) \in S_I$. Since $H$ and $K$ are finite, their union $K \cup H$ must also be finite and included in $I$. Hence $\pi_{H \cup K}(c) \in S_I$.

Therefore, $S_I$ is a directed set.

Since $S = \{\pi_i(c) \mid i \in I\} = \{\pi_{\{i\}}(c) \mid i \in I\}$ is included in $S_I$, we obtain $\bigcup S_I \sqsubseteq \bigcup S \sqsubseteq e$.

But $e$ is compact and $S_I$ directed hence there must be $\pi_I(c) \in S_I$, with $J$ a finite set, such that $\pi_I(c) \sqsubseteq e$. From Prop.24 (3) and Prop.24 (1), we conclude $c \preceq \Delta_J(e)$ as wanted.

Proof of Proposition 27.

(1) There exists a family of scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$ such that $|S(C)| \geq (n - 2)!$ where $n = |Con|$. (2) There exists a family of scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$, with $(Con, \sqsubseteq)$ being a constraint frame, such that $|S(C)| = n^{\log_2 n}$ where $n = |Con|$.

Proof. For (1) take the complete lattice $\mathcal{M}_n$ obtained by adding a top and bottom to the poset $\mathfrak{N}$ obtained by giving $N = \{1, 2, \ldots, n - 2\}$ the discrete order ($x \sqsubseteq y$ if and only if $x = y$). Any function preserving top and bottom and permuting the elements of $N$ is a space function. Hence, there are $(n - 2)!$ such functions. So $|S(C)| \geq (n - 2)!$.

For (2) take the powerset $\mathcal{P}(S)$ of some finite set $S$ ordered by inclusion (join is set union). Suppose $n = |\mathcal{P}(S)|$. Let $F = \{f : \mathcal{P}(S) \rightarrow \mathcal{P}(S)\}$ be the family of functions that satisfy (a) $f(T) = \bigcup_{i \in T} f(i)$ if $|T| > 1$ and (b) $f(\emptyset) = \emptyset$. One can verify that $f$ is a space function. The set $\mathcal{P}(S)$ has $\log_2 n$ singletons. Since there is no restriction on how $f$ should map singletons we conclude $|F| = n^{\log_2 n}$.

Proof of Lemma 28.

Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $C = (Con, \sqsubseteq, (s_i)_{i \in G})$. Suppose that $(Con, \sqsubseteq)$ is a constraint frame. Let $\delta_I : Con \rightarrow Con$, with $I \subseteq G$, be the function

$$\delta_I(c) \overset{\text{def}}{=} \bigcap_{i \in I} \{\bigcup_i s_i(a_i) \mid (a_i)_{i \in I} \in Con^I \text{ and } \bigcup_i a_i \sqsubseteq c\}$$

for every $c \in Con$. Then $\Delta_I = \delta_I^+$. □
Proof. Firstly, we are going to show that $\delta_I^+$ is a group distribution candidate.

(D.1) $\delta_I^+$ is a space function.

1. $\delta_I^+(\text{true}) = \text{true}$.

Note that $\text{true} \in \{ \bigcup_{i \in I} s_i(a_i) \mid \bigcup_{i \in I} a_i \supseteq \text{true} \}$, therefore $\delta_I^+(\text{true}) = \text{true}$.

2. $\delta_I^+(c \sqcup d) = \delta_I^+(c) \sqcup \delta_I^+(d)$.

We prove that $\delta_I^+$ is monotonic: If $c \supseteq d$ then $\delta_I^+(c) \supseteq \delta_I^+(d)$. Assume $c \supseteq d$. If $\bigcup_{i \in I} a_i \supseteq c$, then $\bigcup_{i \in I} a_i \supseteq d$. Therefore, $\{ \bigcup_{i \in I} s_i(a_i) \mid \bigcup_{i \in I} a_i \supseteq c \} \subseteq \{ \bigcup_{i \in I} s_i(a_i) \mid \bigcup_{i \in I} a_i \supseteq d \}$ which implies $\delta_I^+(c) \supseteq \delta_I^+(d)$.

Since $\delta_I^+$ is monotonic, $\delta_I^+(c \sqcup d) \supseteq \delta_I^+(c)$ and $\delta_I^+(c \sqcup d) \supseteq \delta_I^+(d)$, thus $\delta_I^+(c \sqcup d) \supseteq \delta_I^+(c) \sqcup \delta_I^+(d)$.

The other direction follows from this derivation for $(a_i)_{i \in I}, (b_i)_{i \in I}, (c_i)_{i \in I} \in \text{Con}^I$:

\[
\begin{align*}
\delta_I^+(c) \sqcup \delta_I^+(d) &= \langle \text{Definition of } \delta_I^+(d) \rangle \\
\delta_I^+(c) \sqcup \bigcap \left\{ \bigcup_{i \in I} s_i(b_i) \mid \bigcup_{i \in I} b_i \supseteq d \right\} &= \langle \sqcup \text{ distributes over } \sqcap \rangle \\
\bigcap \left\{ \delta_I^+(c) \sqcup \bigcup_{i \in I} s_i(b_i) \mid \bigcup_{i \in I} b_i \supseteq d \right\} &= \langle \text{Definition of } \delta_I^+(c) \rangle \\
\bigcap \left\{ \bigcap \left( \bigcup_{i \in I} s_i(a_i) \mid \bigcup_{i \in I} a_i \supseteq c \right) \cup \bigcup_{i \in I} s_i(b_i) \mid \bigcup_{i \in I} b_i \supseteq d \right\} &= \langle \sqcap \text{ distributes over } \sqcup \rangle \\
\bigcap \left\{ \bigcap \left( \bigcup_{i \in I} s_i(a_i) \cup \bigcup_{i \in I} s_i(b_i) \mid \bigcup_{i \in I} a_i \supseteq c \right) \cup \bigcup_{i \in I} b_i \supseteq d \right\} &= \langle \text{Associativity of } \cap \rangle \\
\bigcap \left\{ \bigcap \left( \bigcup_{i \in I} s_i(a_i) \cup s_i(b_i) \right) \mid \bigcup_{i \in I} a_i \supseteq c \text{ and } \bigcup_{i \in I} b_i \supseteq d \right\} &= \langle x \supseteq y \text{ and } w \supseteq z \text{ implies } x \sqcup w \supseteq y \sqcup z; c_i = a_i \sqcup b_i; s_i(a_i) \sqcup s_i(b_i) = s_i(a_i \sqcup b_i) \rangle \\
\bigcap \left\{ \bigcup_{i \in I} s_i(c_i) \mid \bigcup_{i \in I} c_i \supseteq c \cup d \right\} &= \langle \text{Definition of } \delta_I^+(c \sqcup d) \rangle \\
\delta_I^+(c \sqcup d) &= \langle \text{Definition of } \delta_I^+(c \sqcup d) \rangle \\
\end{align*}
\]

3. $\delta_I^+$ is continuous.

Similar to proof of part (2). Since $(\text{Con}, \sqsubseteq)$ is a finite scs, continuity follows from preservation of finite joins.

(D.2) $\delta_I^+ = s_i$, if $I = \{ i \}$.
Therefore, \( \delta_J^+(c) \subseteq \delta_I^+(c) \) for all \( c \in \text{Con} \).

To complete the proof, we want to show that \( \Delta_I(c) \subseteq \delta_J^+(c) \) for all \( c \in \text{Con} \). Let \( (a_i)_{i \in I} \in \text{Con}^I \) be an arbitrary tuple such that \( \bigcup_{i \in I} a_i \supseteq c \). Since \( \{i\} \subseteq I \) for every \( i \) and \( \Delta_I \) is a gdc, \( \Delta_I(c) \subseteq \Delta_{\{i\}}(c) = s_i(c) \), thus \( \bigcup_{i \in I} \Delta_I(c) \subseteq \bigcup_{i \in I} s_i(c) \). Additionally, \( \bigcup_{i \in I} s_i(c) \subseteq \bigcup_{i \in I} s_i(a_i) \) from monotonicity of every \( s_i \). Therefore, \( \Delta_I(c) \subseteq \bigcup_{i \in I} s_i(a_i) \). Since each \( s_i \) is continuous \( s_i(\bigcup_{i \in I} a_i) = \bigcup_{i \in I} s_i(a_i) \), then \( \Delta_I(c) \subseteq \bigcup_{j \in J} s_j(a_j) \) for any \( (a_i)_{i \in I} \in \text{Con}^I \), thus \( \Delta_I(c) \) is a lower bound of \( \bigcup_{j \in J} s_j(a_j) \) and \( \bigcup_{j \in J} a_j \supseteq c \) and so \( \Delta_I(c) \subseteq \delta_J^+(c) \) for every \( c \in \text{Con} \).

**Proof of Theorem 30.**

Let \( (\Delta_J)_{J \subseteq G} \) be the distributed spaces of a finite scs \( \mathbf{C} = (\text{Con}, \subseteq, (s_i)_{i \in G}) \). Suppose that \( (\text{Con}, \subseteq) \) is a constraint frame. Let \( J, K \subseteq G \) be two groups such that \( I = J \cup K \). Then the following equalities hold:

1. \( \Delta_J(c) = \bigcap \{ \Delta_J(a) \cup \Delta_K(b) \mid a \cup b \supseteq c \} \) \hspace{1cm} (4)
2. \( \Delta_I(c) = \bigcap \{ \Delta_I(a) \cup \Delta_K(a \rightarrow c) \} \) \hspace{1cm} (5)
3. \( \Delta_J(c) = \bigcap \{ \Delta_J(a) \cup \Delta_K(a \rightarrow c) \mid a \subseteq c \} \) \hspace{1cm} (6)

**Proof.** 1. Let \( a, b \in \text{Con} \). From Lemma 28

\[ \Delta_J(a) = \bigcap \{ \bigcup_{j \in J} s_j(a_j) \mid (a_j)_{j \in J} \in \text{Con}^J \text{ and } \bigcup_{j \in J} a_j \supseteq a \} \]
20 Reasoning about Distributed Knowledge

\[ \Delta_K(b) = \bigcap \{ \bigcup_{k \in K} s_k(b_k) \mid (b_k)_{k \in K} \in \text{Con}^K \text{ and } \bigcup_{k \in K} b_k \sqsupseteq b \} \]

Thus, \( \Delta_J(a) \sqcup \Delta_K(b) = \bigcap \{ \bigcup_{i \in I} s_i(c_i) \mid (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigcup_{i \in I} c_i \sqsupseteq a \sqcup b \} \) where for every \( s_j = s_k \), \( s_j(c) = s_j(a_j) \sqcup s_k(b_k) \) and \( c_i = a_i \sqcup b_k \). If \( s_j \neq s_k \), either \( s_j(c_i) = s_j(a_j) \) or \( s_j(c_i) = s_k(b_k) \). Now, given \( c \in \text{Con} \) consider the set \( \{ \Delta_J(a) \sqcup \Delta_K(b) \mid a \sqcup b \sqsupseteq c \} \) for any \( a, b \in \text{Con} \).

\[ \bigcap \{ \Delta_J(a) \sqcup \Delta_K(b) \mid a \sqcup b \sqsupseteq c \} = (\text{Construction of } \Delta_J(a) \sqcup \Delta_K(b)) \]

\[ \bigcap \{ \bigcap_{i \in I} \{ \bigcup_{s_i(c)} (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigcup_{i \in I} c_i \sqsupseteq a \sqcup b \} \mid a \sqcup b \sqsupseteq c \} = (\text{Associativity of } \bigcap) \]

\[ \bigcap \{ \bigcap_{i \in I} \{ \bigcup_{s_i(c)} (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigcup_{i \in I} c_i \sqsupseteq a \sqcup b \} \mid a \sqcup b \sqsupseteq c \} = (\text{Lemma 28}) \]

\[ \Delta_J(c) \]

2. This property can be seen as a simplification of the first one: recall that \( a \rightarrow c \) represents the least element \( c \) such that \( a \sqcup c \sqsupseteq c \). Take any \( b \) such that \( a \sqcup b \sqsupseteq c \). Then \( b \sqsupseteq b \rightarrow c \) and since space functions are monotonic \( \Delta_J(a) \sqcup \Delta_K(b) \sqsupseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \). From this it follows that \( \bigcap \{ S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \} \sqcup \Delta_J(a) \sqcup \Delta_K(b) \} = \bigcap \{ S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \} \} \) for any \( S \subseteq \text{Con} \). This shows that \( \Delta_J(a) \sqcup \Delta_K(b) \) is redundant since \( \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \) is included in the set on the right-hand side of equality in the second property.

3. Similarly, this property can be seen as a simplification of the second one. Take any \( a' \nsubseteq c \).

It suffices to find \( a \subseteq c \) such that \( \Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsupseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \) since then \( \bigcap \{ S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \} \sqcup \Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \} = \bigcap \{ S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \} \} \) for any \( S \subseteq \text{Con} \).

Since \( a' \nsubseteq c \) either \( a' \sqsupseteq c \) or \( a' \) and \( c \) are incomparable w.r.t. \( \sqsubseteq \), written \( a' \parallel c \). Suppose \( a \) holds. Then take \( a = b \) thus \( a \rightarrow c \) is true. By monotonicity we have \( \Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsubseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \) as wanted. Suppose \( a' \parallel c \) holds. Notice that \( a' \rightarrow c \subseteq c \). Suppose that \( a' \rightarrow c = c \). Then we can take \( a = b \), and thus \( a \rightarrow c = a' \rightarrow c \). By monotonicity we have \( \Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsubseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \) as wanted. Suppose \( a' \rightarrow c \subseteq c \) holds. In this case, which is more interesting, we can build a poset \( L = \{ a' \sqcup c, a', c, a' \rightarrow c, a' \rightarrow c \parallel (a' \rightarrow c) \} \sqcap \) and verify that \( L \) is a non-distributive sub-lattice of \( \text{Con} \sqcap \), isomorphic to a lattice known as \( N_5 \) (see Fig. ??). But from order theory we know this cannot happen since we assumed \( (\text{Con}, \sqsubseteq) \) to be distributive, and distributive lattices do not have sub-lattices isomorphic to \( N_5 \) (3).

Proof of Complexity for Algorithms in Section 5.2.

The worst-case time complexity for computing the value \( \Delta_J(c) \) for each algorithm is in (1) \( O(mn^m) \) for \( \Delta_{\text{Part}+} \), (2) \( O(mn^2 \log n) \) for \( \Delta_{\text{Part}1} \), (3) \( O(mn \log n) \) for \( \Delta_{\text{Part}2} \), \( \Delta_{\text{Part}3} \) and \( \Delta_{\text{Part}3+} \).
Proof. (1) \( \text{Delta}^+ \) checks \( n^m \) tuples with a cost \( O(m) \) for a total of \( O(mn^m) \). (2) The worst-case time complexity of \( \text{DeltaPart1} \) is given by the recurrence equation \( T(m) = n^2(1 + 2T(m/2)) \) and \( T(1) = 1 \) whose solution is \( O(mn^2 \log m) \). (3) The worst-case time complexity of \( \text{DeltaPart2}, \text{DeltaPart3} \) and \( \text{DeltaPart3}^+ \) is given by \( T(m) = n(1 + 2T(m/2)) \) and \( T(1) = 1 \) whose solution is \( O(mn^2 \log m) \).

The worst-case time complexity for computing function \( \Delta_I \) with each of the algorithms above is obtained by multiplying the corresponding complexity by a factor of \( n \). ◀

### B Experimental Results

![Figure 4 Experimental results, average time, over \( n \)-element powerset lattices with a randomly generated number of space functions. Note that for the \( x \)-axis 512_8 means an 8-element powerset lattice with 512 nodes, and \( |I| = 8 \).](image)

Figure 4 shows the results of running each of the proposed algorithms to calculate \( \Delta_I \) over a \( n \)-element powerset lattice with a varying number of randomly generated space functions, using Python 3.7.1 on a AC-powered 15-inches MacBook Pro Mid 2014, with an Intel Core i7-4770HQ CPU at 2.2 GHz, and 16 GB 1600 MHz DDR3 RAM. From a 4-element powerset lattice with 16 nodes, to a 10-element powerset lattice with 1024 nodes. For each powerset lattice we generated a set \( \{s_i\}_{i \in I} \) of randomly generated space functions, with \( |I| = 4, 8, 12, 16 \), and then we ran each algorithm multiple times, measured the system-clock time and calculated the average. Missing datapoints in Figure 4 implies that the algorithm took more than 600 seconds, except for \( \text{Delta}^+ \) which took more than 1800 seconds but was kept there for illustration purposes. The same results but only for \( |I| = 4 \) are shown in Table 1.
### Table 1

Average time in seconds for each algorithm over a powerset lattices with $|I| = 4$

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Delta+</th>
<th>DeltaPart1</th>
<th>DeltaPart2</th>
<th>DeltaPart3</th>
<th>DeltaPart3+</th>
<th>DeltaGen</th>
<th>DeltaGen+</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2.01</td>
<td>0.958</td>
<td>0.0135</td>
<td>0.00127</td>
<td>0.000632</td>
<td>0.00360</td>
<td>0.000603</td>
</tr>
<tr>
<td>32</td>
<td>64.6</td>
<td>25.3</td>
<td>0.0990</td>
<td>0.00554</td>
<td>0.00181</td>
<td>0.0633</td>
<td>0.00343</td>
</tr>
<tr>
<td>64</td>
<td>1901</td>
<td>600</td>
<td>0.740</td>
<td>0.0216</td>
<td>0.00542</td>
<td>0.948</td>
<td>0.0154</td>
</tr>
<tr>
<td>128</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>5.82</td>
<td>0.0922</td>
<td>0.0160</td>
<td>15.4</td>
<td>0.0860</td>
</tr>
<tr>
<td>256</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>46.2</td>
<td>0.433</td>
<td>0.0483</td>
<td>252</td>
<td>0.361</td>
</tr>
<tr>
<td>512</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>385</td>
<td>2.31</td>
<td>0.166</td>
<td>&gt;600</td>
<td>2.01</td>
</tr>
<tr>
<td>1024</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>12.5</td>
<td>0.547</td>
<td>&gt;600</td>
<td>10.7</td>
</tr>
</tbody>
</table>

Reasoning about Distributed Knowledge
Index

(Con, ⊑) G, s = (sᵢ)ᵢ∈G, spatial constraint system, scs, 5
Con, set of constraints, 4
C, (Con, ⊑), constraint system, cs, 4
G, group of agents, 5

Δ, distributed spaces, 8
Δ_I, distributed space of a group I, 8
Π_I(c), the I-group projection of c ∈ Con, 10
Δ, entailment, order relation, 3
∼•, Heyting negation, 4
δ, group distribution candidate, 9
false (top), all (possibly inconsistent) information, 3

λ⊥(c), biggest space, bottom element of the lattice, 8
λτ(c), smallest space, empty space, top element of the lattice, 8
S(C), set of all space functions over a cs C, 7
π_I(c), I-join projection of a group I of c, 6
π_i(c), i-agent projection of c ∈ Con, 6
→, Heyting implication, 4
S(C), set of all space functions over C = (Con, ⊑), 5
s, space tuple, 5
sᵢ, space function of agent i, 5
⊔, ⊓, meet, GLB, infimum, 3
⊔, ⊓, join, LUB, supremum, 3

max max operation, 3
ture (bottom), empty information, 3
X^J, denotes the set of tuples (x_j)ₖ∈J of elements x_j ∈ X for each j ∈ J, 12

f ⊑ g, space order, 7
compact element, 3
constraint frame, 4
continuous function, 3

D.1, first axiom group distribution candidate, 9
D.2, second axiom group distribution candidate, 9
D.3, third axiom group distribution candidate, 9
directed set, 3

finite scs, 11
join-homomorphism, 3
join-preservation, 3
lattice, complete, 3
S.1, first space axiom, 5
S.2, second space axiom, 5