Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents

Michell Guzmán, Sophia Knight, Santiago Quintero, Sergio Ramírez, Camilo Rueda, Frank D. Valencia

Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents

Michell Guzmán
University of Milano-Bicocca, Italy
michell.guzman@unimib.it

Sophia Knight
University of Minnesota Duluth
sophia.knight@gmail.com

Santiago Quintero
LIX École Polytechnique de Paris, Francia
squinter@lix.polytechnique.fr

Sergio Ramírez
Pontificia Universidad Javeriana de Cali, Colombia
sergio@javerianacali.edu.co

Camilo Rueda
Pontificia Universidad Javeriana de Cali, Colombia
crueda@javerianacali.edu.co

Frank Valencia
CNRS, LIX École Polytechnique de Paris, Francia
Pontificia Universidad Javeriana de Cali, Colombia
frank.valencia@lix.polytechnique.fr

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.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Distributed computing models; Theory of computation → Semantics and reasoning

Keywords and phrases Reasoning about Groups, Distributed Knowledge, Infinitely Many Agents, Reasoning about Space, Algebraic Modeling.

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.25

Related Version Full version of this paper at `https://hal.archives-ouvertes.fr/hal-02172415`.

Funding This work have been partially supported by the ECOS-NORD project FACTS (C19M03) and the Colciencias project CLASSIC (125171250031).

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

30th International Conference on Concurrency Theory (CONCUR 2019).
Editors: Wan Fokkink and Rob van Glabbeek; Article No. 25; pp. 25:1–25:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 among 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 $(Con, \sqsubseteq)$. 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 $\sqsubseteq$ corresponds to entailment between constraints, $c \sqsubseteq d$, often written $d \sqsupseteq c$, means c can be derived from d, or that d represents as much information as c. The join $\sqcup$, the bottom *true*, and the top *false* of the lattice correspond to conjunction, the empty information, and the join of all (possibly inconsistent) information, respectively.

The notion of computational space and the epistemic notion of belief in the spatial ccp (sccp) process calculi [15] is represented as a family of join-preserving maps $\mathfrak{s}_i : Con \to Con$ called *space functions*. A cs equipped with space functions is called a *spatial constraint system* (scs). From a *computational point of view* $\mathfrak{s}_i(c)$ can be interpreted as an assertion specifying that c resides within the space of agent i. From an *epistemic point of view*, $\mathfrak{s}_i(c)$ specifies that i considers c to be true. An alternative epistemic view is that i interprets c as $\mathfrak{s}_i(c)$. 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 $\langle P, c\rangle$ where P is a process and c is a constraint, called *the store*, representing the current partial information. E.g., a reduction $\langle\ P, \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b)\ \rangle \longrightarrow \langle\ Q, \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b \sqcup c)\ \rangle$ 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.

---

[1] For simplicity we use *cs* for both *constraint system* and its plural form.

Given the above reduction, assume that d is some piece of information resulting from the combination (join) of the three constraints above, i.e., $d = a \sqcup b \sqcup c$, 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 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., $d = \mathit{false}$).

In this work we introduce the theory of group space functions $\Delta_I : \mathit{Con} \to \mathit{Con}$ to reason about information distributed among the members of a potentially infinite group I. We shall refer to Δ_I as the *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 $\mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b \sqcup c) \sqsupseteq \Delta_{\{1,2\}}(d)$ meaning that from the information $\mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b \sqcup c)$ 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 groups can be used to reason about multi-agent computations with unboundedly many agents. For example, a *computation* in sccp is a possibly infinite reduction sequence γ of the form $\langle\ P_0, c_0\ \rangle \longrightarrow \langle\ P_1, c_1\ \rangle \longrightarrow \cdots$ with $c_0 \sqsubseteq c_1 \sqsubseteq \cdots$. The *result* of γ 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 *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, $\mathcal{O}(P, c) \sqsupseteq \Delta_{\mathbb{N}}(\mathit{false})$, where $\mathbb{N}$ 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 $H \subseteq \mathbb{N}$ such that $\mathcal{O}(P, c) \sqsupseteq \Delta_H(\mathit{false})$ implies $\mathcal{O}(P, c) \sqsupseteq \Delta_{\mathbb{N}}(\mathit{false})$. The converse of this implication will be called *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 Δ_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 *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 *compositional* nature: The information of a group is determined by that of its subgroups.
4. We provide a *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 *algorithms* to compute Δ_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 $\mathbf{C}$ be a poset $(Con, \sqsubseteq)$, and let $S \subseteq Con$. We use $\bigsqcup S$ to denote the least upper bound (or *supremum* or *join*) of the elements in S, and $\bigsqcap S$ is the greatest lower 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 $e' \in S$, $e' \sqsubseteq e$. If such e exists, we denote it by $max\ S$. As usual, if $S = \{c, d\}$, $c \sqcup d$ and $c \sqcap d$ represent $\bigsqcup S$ and $\bigsqcap S$, respectively. If $S = \emptyset$, we denote $\bigsqcup S = true$ and $\bigsqcap S = false$. We say that $\mathbf{C}$ is a *complete lattice* iff each subset of Con has a supremum in Con. The poset $\mathbf{C}$ is *distributive* iff for every $a, b, c \in Con$, $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$. A non-empty set $S \subseteq Con$ 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 $c \in Con$ is *compact* iff for any directed subset D of Con, $c \sqsubseteq \bigsqcup D$ implies $c \sqsubseteq d$ for some $d \in D$. A *self-map* on Con is a function f from Con to Con. Let $(Con, \sqsubseteq)$ be a complete lattice. The self-map f on Con *preserves* the join of a set $S \subseteq Con$ iff $f(\bigsqcup S) = \bigsqcup\{f(c) \mid c \in S\}$. A self-map that preserves the join of finite sets is called *join-homomorphism*. A self-map f on 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 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)* $\mathbf{C}$ *is a complete lattice* $(Con, \sqsubseteq)$*. The elements of Con are called* constraints*. The symbols* $\sqcup$*, true and false will be used to denote the least upper bound (lub) operation, the bottom, and the top element of* $\mathbf{C}$*.*

The elements of the lattice, the *constraints*, represent (partial) information. A constraint c 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 $d \sqsupseteq c$ 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* $(Con, \sqsubseteq)$ *is said to be a* constraint frame *iff its joins distribute over arbitrary meets. More precisely,* $c \sqcup \bigsqcap S = \bigsqcap\{c \sqcup e \mid e \in S\}$ *for every* $c \in Con$ *and* $S \subseteq Con$*. Define* $c \rightarrow d$ *as* $\bigsqcap\{e \in Con \mid c \sqcup e \sqsupseteq d\}$*.*

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

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

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 $\mathfrak{s}_i : Con \to Con$ satisfying some structural conditions. Recall that constraints can be viewed as assertions. Thus given $c \in Con$, we can then think of the constraint $\mathfrak{s}_i(c)$ as an assertion stating that c is a piece of information residing *within a space of agent* i. Some alternative *epistemic* interpretations of $\mathfrak{s}_i(c)$ 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 $\mathfrak{s}_i(c)$. All these interpretations convey the idea that c is local or subjective to agent i.

In [15] scs 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 $\langle\ P, \mathfrak{s}_i(c) \sqcup \mathfrak{s}_j(d)\ \rangle \longrightarrow \langle\ Q, \mathfrak{s}_i(c) \sqcup \mathfrak{s}_j(d \sqcup e)\ \rangle$ 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* $(Con, \sqsubseteq)$ *is a* continuous *self-map* $f : Con \to Con$ *s.t. for every* $c, d \in Con$ *(S.1)* $f(true) = true$*, (S.2)* $f(c \sqcup d) = f(c) \sqcup f(d)$*. We shall use* $\mathcal{S}(\mathbf{C})$ *to denote the set of all space functions over* $\mathbf{C} = (Con, \sqsubseteq)$*.*

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 17, that continuity comes naturally in the intended phenomena we wish to capture: modelling information of possibly *infinite* groups. In fact, in [15] scs could only have finitely many agents.

In this work we also extend scs to allow arbitrary, possibly infinite, sets of agents. A *spatial cs* is a cs with a possibly infinite group of agents each having a space function.

▶ **Definition 5** (Spatial Constraint Systems). *A* spatial cs (scs) *is a cs* $\mathbf{C} = (Con, \sqsubseteq)$ *equipped with a possibly infinite tuple* $\mathfrak{s} = (\mathfrak{s}_i)_{i \in G}$ *of space functions from* $\mathcal{S}(\mathbf{C})$*.*

We shall use $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ *to denote an scs with a tuple* $(\mathfrak{s}_i)_{i \in G}$*. We refer to* G *and* $\mathfrak{s}$ *as the* group of agents *and* space tuple *of* $\mathbf{C}$ *and to each* $\mathfrak{s}_i$ *as the* space function *in* $\mathbf{C}$ *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 that will be used throughout the paper.

▶ **Example 6.** The scs $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in \{1,2\}})$ in Fig.1 is given by the complete lattice $\mathbf{M}_2$ and two agents. We have $Con = \{p \vee \neg p, p, \neg p, p \wedge \neg p\}$ and $c \sqsubseteq d$ iff c is a logical consequence of d. The top element *false* is $p \wedge \neg p$, the bottom element *true* is $p \vee \neg p$, and the constraints p and $\neg p$ are incomparable with each other. The set of agents is $\{1, 2\}$ with space functions $\mathfrak{s}_1$ and $\mathfrak{s}_2$: For agent 1, $\mathfrak{s}_1(p) = \neg p$, $\mathfrak{s}_1(\neg p) = p$, $\mathfrak{s}_1(false) = false$, $\mathfrak{s}_1(true) = true$, and for agent 2, $\mathfrak{s}_2(p) = false = \mathfrak{s}_2(false)$, $\mathfrak{s}_2(\neg p) = \neg p$, $\mathfrak{s}_2(true) = true$. 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 Kripke structures and Aumann structures (see [15]). We illustrate scs with infinite groups in the next section.

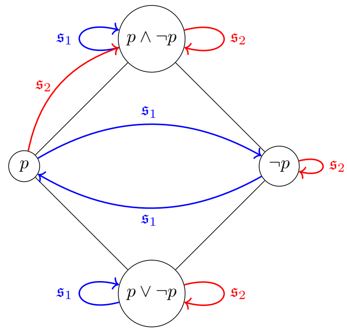


Figure 1 Cs given by lattice $\mathbf{M}_2$ ordered by implication and space functions $\mathfrak{s}_1$ and $\mathfrak{s}_2$.

4 Distributed Information

In this section we characterize the notion of collective information of a group of agents. 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 wish to formalize 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 this section, illustrate the above intuition.

▶ **Example 7.** Consider an scs $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ where $G = \mathbb{N}$ and $(Con, \sqsubseteq)$ is a constraint frame. Let $c \stackrel{\text{def}}{=} \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \rightarrow b) \sqcup \mathfrak{s}_3(b \rightarrow e)$. The 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 necessarily 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 $a \sqcup (a \rightarrow b) \sqcup (b \rightarrow e) \sqsupseteq e$. Let us now consider an example with infinitely many agents. Let $c' \stackrel{\text{def}}{=} \bigsqcup_{i \in \mathbb{N}} \mathfrak{s}_i(a_i)$ for some increasing chain $a_0 \sqsubseteq a_1 \sqsubseteq \dots$. Take e' s.t. $e' \sqsubseteq \bigsqcup_{i \in \mathbb{N}} a_i$. 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 $e' \sqsupset a_i$ 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., $\bigsqcup_{i \in \mathbb{N}} a_i$.

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 8** (Agent and Join Projections). *Let $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be an scs. Given $i \in G$, the i-agent projection of $c \in Con$ is defined as $\pi_i(c) \stackrel{\text{def}}{=} \bigsqcup\{e \mid c \sqsupseteq \mathfrak{s}_i(e)\}$. We say that e is i-agent derivable from c iff $\pi_i(c) \sqsupseteq e$. Given $I \subseteq G$ the I-join projection of a group I of c is defined as $\pi_I(c) \stackrel{\text{def}}{=} \bigsqcup\{\pi_i(c) \mid i \in I\}$. We say that e is I-join derivable from c iff $\pi_I(c) \sqsupseteq e$.*

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.

▶ **Example 9.** Let c be as in Ex.7. We have $\pi_1(c) \sqsupseteq a$, $\pi_2(c) \sqsupseteq (a \rightarrow b)$, $\pi_3(c) \sqsupseteq (b \rightarrow e)$. Indeed e is I-join derivable from c since $\pi_{\{1,2,3\}}(c) = \pi_1(c) \sqcup \pi_2(c) \sqcup \pi_3(c) \sqsupseteq e$. Similarly we conclude that e' is I-join derivable from c' in Ex.7 since $\pi_{\mathbb{N}}(c') = \bigsqcup_{i \in \mathbb{N}} \pi_i(c) \sqsupseteq \bigsqcup_{i \in \mathbb{N}} a_i \sqsupseteq e'$.

Nevertheless, I-join projections do not provide a complete mechanism for reasoning about distributed information as illustrated below.

▶ **Example 10.** Let $d \stackrel{\text{def}}{=} \mathfrak{s}_1(b) \sqcap \mathfrak{s}_2(b)$. Recall that we think of $\sqcup$ and $\sqcap$ as conjunction and disjunction of assertions: d specifies that b is present in the space of agent 1 or in the space of agent 2 though not exactly in which one. Thus from d we should be able to conclude that b belongs to the space of *some* agent in $\{1,2\}$. Nevertheless, in general b is not I-join derivable from d since from $\pi_{\{1,2\}}(d) = \pi_1(d) \sqcup \pi_2(d)$ we cannot, in general, derive b. To see this consider the scs in Fig.2a and take $b = \neg p$. We have $\pi_{\{1,2\}}(d) = \pi_1(d) \sqcup \pi_2(d) = \mathit{true} \sqcup \mathit{true} = \mathit{true} \not\sqsupseteq b$. One can generalize the example to infinitely many agents: Consider the scs in Ex.7. Let $d' \stackrel{\text{def}}{=} \bigsqcap_{i \in \mathbb{N}} \mathfrak{s}_i(b')$. We should be able to conclude from d' that b' is in the space of *some* agent in $\mathbb{N}$ but, in general, b' is not $\mathbb{N}$-join derivable from d'.

4.1 Distributed Spaces

In the previous section we illustrated that the I-join projection of c, $\pi_I(c)$, the join of individual projections, may not project all distributed information of a group I. To solve this problem we shall develop the notion of I-group projection of c, written as $\Pi_I(c)$. To do this we shall first define a space function Δ_I called the distributed space of group I. The function Δ_I can be thought of as a virtual space including all the information that can be in the space of a member of I. We shall then define an I-projection Π_I in terms of Δ_I much like π_i is defined in terms of $\mathfrak{s}_i$.

Recall that $\mathcal{S}(\mathbf{C})$ denotes the set of all space functions over a cs $\mathbf{C}$. For notational convenience, we shall use $(f_I)_{I \subseteq G}$ to denote the tuple $(f_I)_{I \in \mathcal{P}(G)}$ of elements of $\mathcal{S}(\mathbf{C})$.

Set of Space Functions. We begin by introducing a new partial order induced by $\mathbf{C}$. The set of space functions ordered point-wise.

▶ **Definition 11** (Space Functions Order). *Let $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be an scs. Given $f, g \in \mathcal{S}(\mathbf{C})$, define $f \sqsubseteq_s g$ iff $f(c) \sqsubseteq g(c)$ for every $c \in Con$. We shall use $\mathbf{C}_s$ to denote the partial order $(\mathcal{S}(\mathbf{C}), \sqsubseteq_s)$; the set of all space functions ordered by $\sqsubseteq_s$.*

A very important fact for the design of our structure is that the set of space functions $\mathcal{S}(\mathbf{C})$ can be made into a complete lattice.

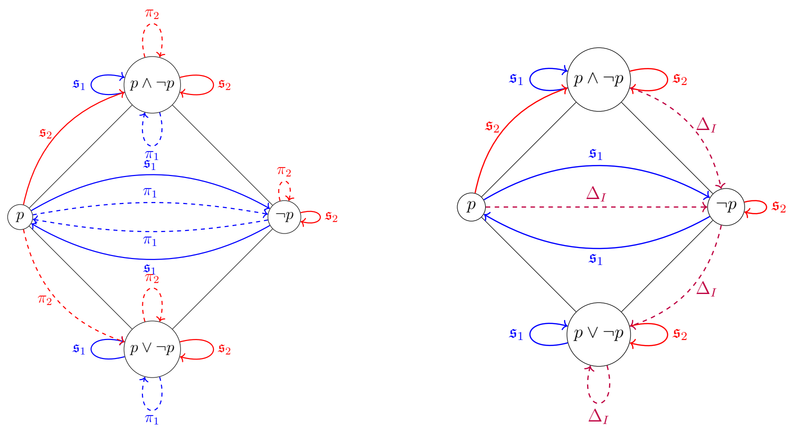
▶ **Lemma 12.** *Let $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be an scs. Then $\mathbf{C}_s$ is a complete lattice.*

4.2 Distributed Spaces as Maximum Spaces.

Let us consider the lattice of space functions $\mathbf{C}_\mathsf{s} = (\mathcal{S}(\mathbf{C}), \sqsubseteq_\mathsf{s})$. Suppose that f and g are space functions in $\mathbf{C}_\mathsf{s}$ with $f \sqsubseteq_\mathsf{s} g$. Intuitively, every piece of information c in the space represented by g is also in the space represented by f since $f(c) \sqsubseteq g(c)$ for every $c \in Con$. 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 $\mathbf{C}_\mathsf{s}$.

Following the above intuition, the order relation $\sqsubseteq_\mathsf{s}$ of $\mathbf{C}_\mathsf{s}$ represents (reverse) space inclusion and the join and meet operations in $\mathbf{C}_\mathsf{s}$ represent intersection and union of spaces. The biggest and the smallest spaces are represented by the bottom and the top elements of the lattice $\mathbf{C}_\mathsf{s}$, here called $\lambda_\perp$ and $\lambda_\top$ and defined as follows.

▶ **Definition 13** (Top and Bottom Spaces). *For every $c \in Con$, define $\lambda_\perp(c) \stackrel{\text{def}}{=} \mathit{true}$, $\lambda_\top(c) \stackrel{\text{def}}{=} \mathit{true}$ if $c = \mathit{true}$ and $\lambda_\top(c) \stackrel{\text{def}}{=} \mathit{false}$ if $c \neq \mathit{true}$.*



(a) Projections π_1 and π_2 given $\mathfrak{s}_1$ and $\mathfrak{s}_2$. **(b)** Δ_I with $I = \{1, 2\}$ given $\mathfrak{s}_1$ and $\mathfrak{s}_2$.

Figure 2 Projections (a) and Distributed Space function (b) over lattice $\mathbf{M}_2$.

The distributed space Δ_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, Δ_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 $(\mathcal{S}(\mathbf{C}), \sqsubseteq_{\mathsf{s}})$ (Lemma 12).

▶ **Definition 14** (Distributed Spaces). *Let* $\mathbf{C}$ *be an scs* $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. *The* distributed spaces *of* $\mathbf{C}$ *is given by* $\Delta = (\Delta_I)_{I \subseteq G}$ *where* $\Delta_I \stackrel{\text{def}}{=} max\{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq_s \mathfrak{s}_i \text{ for every } i \in I\}$. *We shall say that* e *is distributed among* $I \subseteq G$ *w.r.t.* c *iff* $c \sqsupseteq \Delta_I(e)$. *We shall refer to each* Δ_I *as the* (distributed) space *of the group* I.

It follows from Lemma 12 that $\Delta_I = \bigsqcap\{\mathfrak{s}_i \mid i \in I\}$ (where $\bigsqcap$ is the meet in the complete lattice $(\mathcal{S}(\mathbf{C}), \sqsubseteq_{\mathsf{s}})$). Fig.2b illustrates an 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 the distributive information of its subgroups.

▶ **Theorem 15.** *Let* $(\Delta_I)_{I \subseteq G}$ *be the distributed spaces of an scs* $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. *Suppose that* $K, J \subseteq I \subseteq G$. *(1)* $\Delta_I = \lambda_\top$ *if* $I = \emptyset$, *(2)* $\Delta_I = \mathfrak{s}_i$ *if* $I = \{i\}$, *(3)* $\Delta_J(a) \sqcup \Delta_K(b) \sqsupseteq \Delta_I(a \sqcup b)$, *and (4)* $\Delta_J(a) \sqcup \Delta_K(a \to c) \sqsupseteq \Delta_I(c)$ *if* $(Con, \sqsubseteq)$ *is a constraint frame.*

Recall that $\lambda_\top$ corresponds to the empty space (see Def.13). 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 $c \sqsupseteq \Delta_\emptyset(e)$ and $c \neq \mathit{false}$ then $e = \mathit{true}$. Intuitively, the second property says that the function Δ_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.15.

▶ **Example 16.** Let $c = \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \to b) \sqcup \mathfrak{s}_3(b \to e)$ as in Ex.7. We want to prove that e is distributed among $I = \{1, 2, 3\}$ w.r.t. c, i.e., $c \sqsupseteq \Delta_{\{1,2,3\}}(e)$. Using Properties 2 and 4 in Thm.15 we obtain $c \sqsupseteq \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \to b) = \Delta_{\{1\}}(a) \sqcup \Delta_{\{2\}}(a \to b) \sqsupseteq \Delta_{\{1,2\}}(b)$, and then $c \sqsupseteq \Delta_{\{1,2\}}(b) \sqcup \mathfrak{s}_3(b \to e) = \Delta_{\{1,2\}}(b) \sqcup \Delta_{\{3\}}(b \to e) \sqsupseteq \Delta_{\{1,2,3\}}(e)$ as wanted.

▶ Remark 17 (Continuity). The example with infinitely many agents in Ex.7 illustrates well why we require our spaces to be continuous in the presence of possibly infinite groups. Clearly $c' = \bigsqcup_{i\in\mathbb{N}} \mathfrak{s}_i(a_i) \sqsupseteq \bigsqcup_{i\in\mathbb{N}} \Delta_{\mathbb{N}}(a_i)$. By continuity, $\bigsqcup_{i\in\mathbb{N}} \Delta_{\mathbb{N}}(a_i) = \Delta_{\mathbb{N}}(\bigsqcup_{i\in\mathbb{N}} a_i)$ which indeed captures the idea that each a_i is in the distributed space $\Delta_{\mathbb{N}}$.

In Thm.15 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 from mathematical economics: Aumann structures. We illustrate that the notion of distributed knowledge in these structures is an instance of a distributed space.

▶ **Example 18.** *Aumann Constraint Systems.* Aumann structures [13] are an *event-based* approach to modelling knowledge. An Aumann structure is a tuple $\mathcal{A} = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ where S is a set of states and each $\mathcal{P}_i$ is a partition on S for agent i. The partitions are called *information sets*. If two states t and u are in the same information set for agent i, it means that in state t agent i considers state u possible, and vice versa. An *event* in an Aumann structure is any subset of S. Event e holds at state t if $t \in e$. The set $\mathcal{P}_i(s)$ denotes the information set of $\mathcal{P}_i$ containing s. The event of *agent i knowing e* is defined as $\mathsf{K}_i(e) = \{s \in S \mid \mathcal{P}_i(s) \subseteq e\}$, and the *distributed knowledge of an event e among the agents in a group I* is defined as $\mathsf{D}_I(e) = \{s \in S \mid \bigcap_{i\in I} \mathcal{P}_i(s) \subseteq e\}$.

An Aumann structure can be seen as a spatial constraint system $\mathbf{C}(\mathcal{A})$ with events as constraints, i.e., $Con = \{e \mid e \text{ is an event in } \mathcal{A}\}$, and for every $e_1, e_2 \in Con$, $e_1 \sqsubseteq e_2$ iff $e_2 \subseteq e_1$. The operators join ($\sqcup$) and meet ($\sqcap$) are intersection ($\cap$) and union ($\cup$) of events, respectively; $true = S$ and $false = \emptyset$. The space functions are the knowledge operators, i.e., $\mathfrak{s}_i(c) = \mathsf{K}_i(c)$. From these definitions and since meets are unions one can easily verify that $\Delta_I(c) = \mathsf{D}_I(c)$ which shows the correspondence between distributed information and distributed knowledge.

4.3 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.

▶ **Definition 19** (Distribution Candidates). *Let $\mathbf{C}$ be an scs $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$. 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 $I, J \subseteq G$: (D.1) δ_I is a space function in $\mathbf{C}$, (D.2) $\delta_I = \mathfrak{s}_i$ if $I = \{i\}$, (D.3) $\delta_I \sqsupseteq_s \delta_J$ if $I \subseteq J$.*

Property D.1 requires each δ_I to be a space function. This is trivially met for $\delta_I = \Delta_I$. Property D.2 says that the function δ_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 $\delta_I(c) \sqsupseteq \delta_J(c)$, 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.

Properties D1-D3, however, do not determine Δ uniquely. In fact, there could be infinitely-many tuples of space functions that satisfy them. For example, if we were to chose $\delta_\emptyset = \lambda_\top$, $\delta_{\{i\}} = \mathfrak{s}_i$ for every $i \in G$, and $\delta_I = \lambda_\bot$ 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 20** (Max gdc). *Let $(\Delta_I)_{I\subseteq G}$ be the distributed spaces of $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$. Then $(\Delta_I)_{I\subseteq G}$ is a gdc of $\mathbf{C}$ and if $(\delta_I)_{I\subseteq G}$ is a gdc of $\mathbf{C}$ then $\delta_I \sqsubseteq_s \Delta_I$ for each $I \subseteq G$.*

Let us illustrate the use of Properties D1-D3 in Thm.20 with the following example.

▶ **Example 21.** Let $c = \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \to b) \sqcup \mathfrak{s}_3(b \to e)$ as in Ex.7. We want to prove $c \sqsupseteq \Delta_I(e)$ for $I = \{1, 2, 3\}$. From D.2 we have $c = \Delta_{\{1\}}(a) \sqcup \Delta_{\{2\}}(a \to b) \sqcup \Delta_{\{3\}}(b \to e)$. We can then use D.3 to obtain $c \sqsupseteq \Delta_I(a) \sqcup \Delta_I(a \to b) \sqcup \Delta_I(b \to e)$. Finally, by D.1 and Proposition 3 we infer $c \sqsupseteq \Delta_I(a \sqcup (a \to b) \sqcup (b \to e)) \sqsupseteq \Delta_I(e)$, thus $c \sqsupseteq \Delta_I(e)$ as wanted. Now consider our counter-example in Ex.10 with $d = \mathfrak{s}_1(b) \sqcap \mathfrak{s}_2(b)$. 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 $d \sqsupseteq d' = \Delta_{\{1\}}(b) \sqcap \Delta_{\{2\}}(b) \sqsupseteq \Delta_{\{1,2\}}(b) \sqcap \Delta_{\{1,2\}}(b) = \Delta_{\{1,2\}}(b)$ as wanted.

The characterization of distributed spaces by Thm.20 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 $f_I \sqsupseteq_s \Delta_I$ 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 22** (Group Projection). *Let $(\Delta_I)_{I\subseteq G}$ be the distributed spaces of an scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$. Given the set $I \subseteq G$, the I-group projection of $c \in Con$ is defined as $\Pi_I(c) \stackrel{\text{def}}{=} \bigsqcup\{e \mid c \sqsupseteq \Delta_I(e)\}$. 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 23.** *Let $(\Delta_I)_{I\subseteq G}$ be the distributed spaces of $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$. For every $c, e \in Con$, (1) $c \sqsupseteq \Delta_I(e)$ iff $\Pi_I(c) \sqsupseteq e$, (2) $\Pi_I(c) \sqsupseteq \Pi_J(c)$ if $J \subseteq I$, and (3) $\Pi_I(c) \sqsupseteq \pi_I(c)$.*

The first property in Prop.23, 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.10.

4.5 Group Compactness.

Suppose that an *infinite* group of agents I can derive e from c (i.e., $c \sqsupseteq \Delta_I(e)$). 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 24** (Group Compactness). *Let $(\Delta_I)_{I\subseteq G}$ be the distributed spaces of an scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$. Suppose that $c \sqsupseteq \Delta_I(e)$. If e is compact and I-join derivable from c then there exists a finite set $J \subseteq I$ such that $c \sqsupseteq \Delta_J(e)$.*

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

▶ **Example 25.** Consider the example with infinitely many agents in Ex.7. We have $c' = \bigsqcup_{i\in\mathbb{N}} \mathfrak{s}_i(a_i)$ for some increasing chain $a_0 \sqsubseteq a_1 \sqsubseteq \ldots$ and e' s.t. $e' \sqsubseteq \bigsqcup_{i\in\mathbb{N}} a_i$. Notice that $c' \sqsupseteq \Delta_{\mathbb{N}}(e')$ and $\pi_{\mathbb{N}}(c') \sqsupseteq e'$. Hence e' is $\mathbb{N}$-join derivable from c'. If e' is compact, by Thm.24 there must be a finite subset $J \subseteq \mathbb{N}$ such that $c' \sqsupseteq \Delta_J(e')$.

5 Computing Distributed Information

Let us consider a *finite* scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i\in G})$ with distributed spaces $(\Delta_I)_{I\subseteq G}$. By finite scs we mean that Con and G are finite sets. Let us consider the problem of computing Δ_I: Given a set $\{\mathfrak{s}_i\}_{i\in I}$ of space functions, we wish to find the greatest space function f such that $f \sqsubseteq \mathfrak{s}_i$ for all $i \in I$ (see Def.14).

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 $S = \{\mathfrak{s}_1, \mathfrak{s}_2\}$ 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.14 and Lemma 12 that Δ_I equals the following

$$max\{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq \mathfrak{s}_i \text{ for all } i \in I\} = \bigsqcup\{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq \mathfrak{s}_i \text{ for all } i \in I\} = \bigsqcap\{\mathfrak{s}_i \mid i \in I\}$$

A *naive (meet-based) approach* would be to compute $\Delta_I(c)$ by taking the point-wise meet construction $\sigma_I(c) \stackrel{\text{def}}{=} \bigsqcap\{\mathfrak{s}_i(c) \mid i \in I\}$ for each $c \in Con$. But this does not work in general since $\Delta_I(c) = \bigsqcap\{\mathfrak{s}_i \mid i \in I\}(c)$ is not necessarily equal to $\sigma_I(c) = \bigsqcap\{\mathfrak{s}_i(c) \mid i \in I\}$. In fact $\sigma_I \sqsupseteq_{\mathsf{s}} \Delta_I$ but σ_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 $\{f(c) \mid f \in \mathcal{S}(\mathbf{C})$ and $f \sqsubseteq \mathfrak{s}_i$ 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: $(\bigsqcup S)(c) = \bigsqcup\{f(c) | f \in S\}$. However, the number of such functions in $\mathcal{S}(\mathbf{C})$ can be at least factorial in the size of Con. For constraint frames, which under the finite assumption coincides with distributive lattices, the size of $\mathcal{S}(\mathbf{C})$ can be non-polynomial in the size of Con.

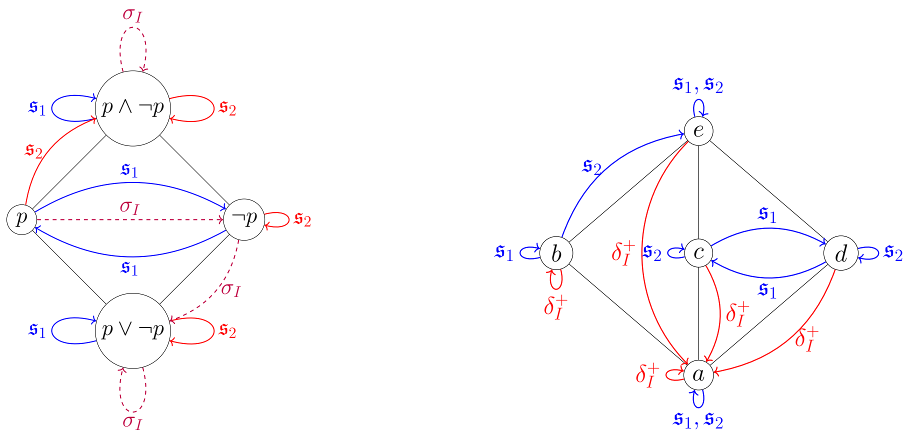
▶ **Proposition 26** (Lower Bounds on Number of Space Functions)**.** *For every $n \geq 2$, there exists a cs $\mathbf{C} = (Con, \sqsubseteq)$ such that $|\mathcal{S}(\mathbf{C})| \geq (n-2)!$ and $n = |Con|$. For every $n \geq 1$, there exists a constraint frame $\mathbf{C} = (Con, \sqsubseteq)$ such that $|\mathcal{S}(\mathbf{C})| \geq n^{\log_2 n}$ and $n = |Con|$.*

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 $c \in Con$ in polynomial time in the size of Con. The first approach uses the inherent compositional nature of Δ_I in distributed lattices. The second approach uses the above-mentioned σ as a suitable upper bound to compute Δ_I 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 Δ_I in distributed lattices (constraint frames). These properties 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 $j \in J$.



(a) For $I = \{1,2\}$, $\sigma_I(c) = \bigsqcap_{i \in I} \mathfrak{s}_i(c)$ is not a space function: $\sigma_I(p \sqcup \neg p) \neq \sigma_I(p) \sqcup \sigma_I(\neg p)$.

(b) For $I = \{1,2\}$, δ_I^+ (Lemma 27) is not a space function: $\delta_I^+(b) \sqcup \delta_I^+(e) = b \neq a = \delta_I^+(b \sqcup e)$.

Figure 3 Counter-examples over lattice $\mathbf{M}_2$ (a) and the non- distributive lattice $\mathbf{M}_3$ (b).

▶ **Lemma 27.** *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that $(Con, \sqsubseteq)$ is a constraint frame. Let $\delta_I^+ : Con \to Con$, with $I \subseteq G$, be the function $\delta_I^+(c) \stackrel{\texttt{def}}{=} \bigsqcap \{ \bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid (a_i)_{i \in I} \in Con^I$ and $\bigsqcup_{i \in I} a_i \sqsupseteq c\}$. 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 $\delta_I^+ \sqsupseteq_{\mathsf{s}} \Delta_I$ uses the fact that space functions preserve joins. The proof that $\delta_I^+ \sqsubseteq_{\mathsf{s}} \Delta_I$ proceeds by showing that $(\delta_I^+)_{I \subseteq G}$ is a group distribution candidate (Def.19). Distributivity of the lattice $(Con, \sqsubseteq)$ 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 28.** Consider the non-distributive lattice $\mathbf{M}_3$ and the space functions $\mathfrak{s}_1$ and $\mathfrak{s}_2$ in Fig.3b. We obtain $\delta_I^+(b \sqcup c) = \delta_I^+(e) = a$ and $\delta_I^+(b) \sqcup \delta_I^+(c) = b \sqcup a = b$. Then, $\delta_I^+(b \sqcup c) \neq \delta_I^+(b) \sqcup \delta_I^+(c)$, i.e., δ_I^+ is not a space function.

Lemma 27 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 29.** *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that $(Con, \sqsubseteq)$ is a constraint frame. Let $J, K \subseteq G$ be two groups such that $I = J \cup K$. Then the following equalities hold:*

1. $$\Delta_I(c) = \bigsqcap \{ \Delta_J(a) \sqcup \Delta_K(b) \mid a, b \in Con \text{ and } a \sqcup b \sqsupseteq c\}. \tag{1}$$
2. $$\Delta_I(c) = \bigsqcap \{ \Delta_J(a) \sqcup \Delta_K(a \to c) \mid a \in Con\}. \tag{2}$$
3. $$\Delta_I(c) = \bigsqcap \{ \Delta_J(a) \sqcup \Delta_K(a \to c) \mid a \in Con \text{ and } a \sqsubseteq c\}. \tag{3}$$

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.29 essentially reformulates Lemma 27 in terms of subgroups rather than agents. It can be proven by replacing $\Delta_J(a)$ and $\Delta_K(b)$ by $\delta_J^+(a)$ and $\delta_K^+(b)$, defined in Lemma 27 and using distributivity of joins over meets. The second and third properties in Thm.29 are pleasant simplifications of the first using heyting implication. These properties realize the intuition that by joining the information a and $a \rightarrow c$ 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.13). Given a finite scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ with distributed spaces $(\Delta_I)_{I \subseteq G}$, the recursive function $\textsc{DeltaPart3}(I, c)$ in Algorithm 1 computes $\Delta_I(c)$ for any given $c \in Con$. Its correctness, assuming that $(Con, \sqsubseteq)$ is a constraint frame (i.e., a distributed lattice), follows from Thm.29(3). Termination follows from the finiteness of $\mathbf{C}$ 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 $I = J \cup K$ to avoid significant recalculation.

Algorithm 1 Function $\textsc{DeltaPart3}(I, c)$ computes $\Delta_I(c)$

```
1: function DeltaPart3(I, c)
2:     if I = ∅ then
3:         return λ⊤(c)
4:     else if I = {i} then
5:         return 𝔰_i(c)
6:     else
7:         {J, K} ← Partition(I)        ▷ returns a partition {J, K} of I s.t., |J| = ⌊|I|/2⌋
8:         return ⨅{DeltaPart3(J, a) ⊔ DeltaPart3(K, a → c) | a ∈ Con and a ⊑ c}.
```

Algorithms. $\textsc{DeltaPart3}(I, c)$ computes $\Delta_I(c)$ using Thm.29(3). By modifying Line 8 with the corresponding meet operations, we obtain two variants of $\textsc{DeltaPart3}$ that use, instead of Thm.29(3), the Properties Thm.29(1) and Thm.29(2). We call them $\textsc{DeltaPart1}$ and $\textsc{DeltaPart2}$. Finally, we also obtain a non-recursive algorithm that outputs $\Delta_I(c)$ by computing $\delta_I^+(c)$ in Lemma 27 in the obvious way: Computing the meet of elements of the form $\bigsqcup_{i \in I} \mathfrak{s}_i(a_i)$ for every tuple $(a_i)_{i \in I}$ such that $\bigsqcup_{i \in I} a_i \sqsupseteq c$. We call it $\textsc{Delta+}$.

Worst-case time complexity. We assume that binary distributive lattice operations $\sqcap$, $\sqcup$, and $\rightarrow$ are computed in $O(1)$ time. We also assume a fixed group I of size $m = |I|$ and express the time complexity for computing Δ_I in terms of $n = |Con|$, 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 Δ_I is in (1) $O(mn^{1+m})$ using $\textsc{Delta+}$, (2) $O(mn^{1+2\log_2 m})$ using $\textsc{DeltaPart1}$, and (3) $O(mn^{1+\log_2 m})$ using $\textsc{DeltaPart2}$ and $\textsc{DeltaPart3}$.

5.3 Algorithm for Arbitrary Lattices

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

There are two ways of correcting σ values: (1) when $\sigma(u) \sqcup \sigma(v) \sqsubset \sigma(u \sqcup v)$, assign $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$ and (2) when $\sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(u \sqcup v)$, assign $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$

and also $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$. It can be shown that the assignments in both cases should decrease σ while preserving the $\sigma \sqsupseteq \Delta_I$ invariant.

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

Algorithm 2 DeltaGen finds Δ_I

```
σ(u) ← ⨅{𝔰_i(u) | i ∈ I}                              ▷ for all u ∈ Con
while u, v ∈ Con ∧ σ(u) ⊔ σ(v) ≠ σ(u ⊔ v) do
    if σ(u) ⊔ σ(v) ⊏ σ(u ⊔ v) then                     ▷ case (1)
        σ(u ⊔ v) ← σ(u) ⊔ σ(v)
    else                                               ▷ case (2)
        σ(u) ← σ(u) ⊓ σ(u ⊔ v)
        σ(v) ← σ(v) ⊓ σ(u ⊔ v)
```

Assume a fixed group I of size $m = |I|$ and that $\sqcap$ and $\sqcup$ are computed in $O(1)$ time. The complexity of the initialization of DeltaGen is $O(nm)$ with $n = |\mathit{Con}|$. 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

1 Samson Abramsky and Achim Jung. Domain theory. In Samson Abramsky et al., editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Oxford University Press, 1994.

2 Frank S. Boer, Alessandra Di Pierro, and Catuscia Palamidessi. Nondeterminism and infinite computations in constraint programming. *Theoretical Computer Science*, pages 37–78, 1995.

3 Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2nd edition, 2002.

4 Joan-María Esteban and Debraj Ray. On the measurement of polarization. *Econometrica*, 62(4):819–851, 1994.

5 Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about knowledge*. MIT press Cambridge, 4th edition, 1995.

6 Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael Mislove, and Dana S. Scott. *Continuous lattices and domains*. Cambridge University Press, 2003.

7 Michell Guzmán, Stefan Haar, Salim Perchy, Camilo Rueda, and Frank Valencia. Belief, Knowledge, Lies and Other Utterances in an Algebra for Space and Extrusion. *Journal of Logical and Algebraic Methods in Programming*, 2016. URL: `https://hal.inria.fr/hal-01257113`.

8 Michell Guzman, Salim Perchy, Camilo Rueda, and Frank Valencia. Deriving Inverse Operators for Modal Logic. In *Theoretical Aspects of Computing – ICTAC 2016*, volume 9965 of *Lecture Notes in Computer Science*, pages 214–232. Springer, 2016. URL: `https://hal.inria.fr/hal-01328188`.

9 Michell Guzmán, Salim Perchy, Camilo Rueda, and Frank Valencia. Characterizing Right Inverses for Spatial Constraint Systems with Applications to Modal Logic. *Theoretical Computer Science*, 744(56–77), October 2018. URL: `https://hal.inria.fr/hal-01675010`.

10 Stefan Haar, Salim Perchy, Camilo Rueda, and Frank Valencia. An Algebraic View of Space/Belief and Extrusion/Utterance for Concurrency/Epistemic Logic. In *17th International Symposium on Principles and Practice of Declarative Programming (PPDP 2015)*, pages 161–172. ACM SIGPLAN, 2015. URL: `https://hal.inria.fr/hal-01256984`.

11 Joseph Y Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual review of computer science*, 2(1):37–68, 1987.

12 Joseph Y Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM (JACM)*, 37(3):549–587, 1990.

13 Joseph Y Halpern and Richard A Shore. Reasoning about common knowledge with infinitely many agents. *Information and Computation*, 191(1):1–40, 2004.

14 Werner Hildenbrand. On economies with many agents. *Journal of Economic Theory*, 2(2):161 – 188, 1970.

15 Sophia Knight, Catuscia Palamidessi, Prakash Panangaden, and Frank D. Valencia. Spatial and Epistemic Modalities in Constraint-Based Process Calculi. In *CONCUR 2012 - 23rd International Conference on Concurrency Theory*, volume 7454 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2012. URL: `https://hal.archives-ouvertes.fr/hal-00761116`.

16 Sophia Knight, Prakash Panangaden, and Frank Valencia. Computing with epistemic and spatial modalities. Submitted for journal publication, 2019.

17 Camilo Rueda and Frank Valencia. On validity in modelization of musical problems by ccp. *Soft Computing*, 8(9):641–648, 2004. URL: `https://doi.org/10.1007/s00500-004-0390-7`.

18 Vijay A. Saraswat, Martin Rinard, and Prakash Panangaden. The semantic foundations of concurrent constraint programming. In *Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '91, pages 333–352. ACM, 1991. URL: `http://doi.acm.org/10.1145/99583.99627`.