



HAL
open science

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. 30th International Conference on Concurrency Theory (CONCUR 2019), Aug 2019, Amsterdam, Netherlands. 10.4230/LIPIcs.CONCUR.2019.29 . hal-02172415v1

HAL Id: hal-02172415

<https://hal.science/hal-02172415v1>

Submitted on 3 Jul 2019 (v1), last revised 6 Nov 2019 (v2)

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.

Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents

Michell Guzmán

University of Milano-Bicocca, Italy
michellrad@gmail.com

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

Keywords and phrases Spatial Information, Distributed Knowledge, Infinitely Many Agents.

Acknowledgements The last fourth authors have been partially supported by the ECOS-NORD project FACTS (C19M03) and 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 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.

45 In this paper we pursue the development of a principled account of a reified notion of
 46 group by taking inspiration from the epistemic notion of *distributed knowledge* [12]. A group
 47 has its information distributed among its member agents. We thus develop a theory about
 48 what exactly is the information available to agents as a group when considering all that is
 49 distributed amongst its members.

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

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

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

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

74 The notion of computational space and the epistemic notion of belief in the spatial ccp
 75 (sccp) process calculi [15] is represented as a family of join-preserving maps $\mathfrak{s}_i : Con \rightarrow Con$
 76 called *space functions*. A cs equipped with space functions is called a *spatial constraint*
 77 *system* (scs). From a *computational point of view* $\mathfrak{s}_i(c)$ can be interpreted as an assertion
 78 specifying that c resides within the space of agent i . From an *epistemic point of view*, $\mathfrak{s}_i(c)$
 79 specifies that i considers c to be true. An alternative epistemic view is that i interprets c as
 80 $\mathfrak{s}_i(c)$. All these interpretations convey the idea of c being local or subjective to agent i .

81 *This work.* In the spatial ccp process calculus *sccp* [15], scs are used to specify the
 82 spatial distribution of information in configurations $\langle P, c \rangle$ where P is a process and c is a
 83 constraint, called *the store*, representing the current partial information. E.g., a reduction
 84 $\langle P, \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b) \rangle \longrightarrow \langle Q, \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(b \sqcup c) \rangle$ means that P , with a in the space of agent 1
 85 and b in the space of agent 2, can evolve to Q while adding c to the space of agent 2.

86 Given the above reduction, assume that d is some piece of information resulting from the
 87 combination (join) of the three constraints above, i.e., $d = a \sqcup b \sqcup c$, but strictly above the join
 88 of any two of them. We are then in the situation where neither agent has d in their spaces,
 89 but as a group they could potentially have d by combining their information. Intuitively, d is
 90 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.

91 agents 1 and 2 may derive as group is a relevant issue in multi-agent concurrent systems,
 92 particularly if d represents unwanted or conflicting information (e.g., $d = false$).

93 In this work we introduce the theory of group space functions $\Delta_I : Con \rightarrow Con$ to reason
 94 about information distributed among the members of a potentially infinite group I . We shall
 95 refer to Δ_I as the *distributed space* of group I . In our theory $c \sqsupseteq \Delta_I(e)$ holds exactly when
 96 we can derive from c that e is distributed among the agents in I . E.g., for d above, we should
 97 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
 98 can derive that d is distributed among the group $I = \{1, 2\}$. Furthermore, $\Delta_I(e) \sqsupseteq \Delta_J(e)$
 99 holds whenever $I \subseteq J$ since if e is distributed among a group I , it should also be distributed
 100 in a group that includes the agents of I .

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

116 Our main contributions are listed below.

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

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

133 2 Background

134 We presuppose basic knowledge of domain and order theory [3, 1, 6] and use the following
 135 notions. Let \mathbf{C} be a poset (Con, \sqsubseteq) , and let $S \subseteq Con$. We use $\bigsqcup S$ to denote the least
 136 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 $\bigsqcup S = \text{true}$ and $\bigsqcap S = \text{false}$. 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 $a, b, c \in \text{Con}$, $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$. 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 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 $(\text{Con}, \sqsubseteq)$ be a complete lattice. The self-map f on Con *preserves* the join of a set $S \subseteq \text{Con}$ 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 $(\text{Con}, \sqsubseteq)$. 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 $d \supseteq c$, 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 $c \neq \text{false}$, 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, $c \sqcup \bigsqcap S = \bigsqcap \{c \sqcup e \mid e \in S\}$ for every $c \in \text{Con}$ and $S \subseteq \text{Con}$. Define $c \rightarrow d$ as $\bigsqcap \{e \in \text{Con} \mid c \sqcup e \supseteq d\}$.*

The following properties of Heyting implication correspond to standard logical properties (with \rightarrow , \sqcup , and \supseteq 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 $c \supseteq d$.*

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.

181 A group G is a set of agents. Each $i \in G$ has a *space function* $\mathfrak{s}_i : \text{Con} \rightarrow \text{Con}$ satisfying
 182 some structural conditions. Recall that constraints can be viewed as assertions. Thus given
 183 $c \in \text{Con}$, we can then think of the constraint $\mathfrak{s}_i(c)$ as an assertion stating that c is a piece of
 184 information residing *within a space of agent i* . Some alternative *epistemic* interpretations of
 185 $\mathfrak{s}_i(c)$ is that it is an assertion stating that agent i *believes* c , that c holds within the space of
 186 agent i , or that agent i *interprets* c as $\mathfrak{s}_i(c)$. All these interpretations convey the idea that c
 187 is local or subjective to agent i .

188 In [15] scs are used to specify the spatial distribution of information in configurations
 189 $\langle P, c \rangle$ where P is a process and c is a constraint. E.g., a reduction $\langle P, \mathfrak{s}_i(c) \sqcup \mathfrak{s}_j(d) \rangle \longrightarrow$
 190 $\langle Q, \mathfrak{s}_i(c) \sqcup \mathfrak{s}_j(d \sqcup e) \rangle$ means that P with c in the space of agent i and d in the space of
 191 agent j can evolve to Q while adding e to the space of agent j .

192 We now introduce the notion of space function.

193 **► Definition 4 (Space Functions).** A space function over a cs $(\text{Con}, \sqsubseteq)$ is a continuous self-
 194 map $f : \text{Con} \rightarrow \text{Con}$ s.t. for every $c, d \in \text{Con}$ (S.1) $f(\text{true}) = \text{true}$, (S.2) $f(c \sqcup d) = f(c) \sqcup f(d)$.
 195 We shall use $\mathcal{S}(\mathbf{C})$ to denote the set of all space functions over $\mathbf{C} = (\text{Con}, \sqsubseteq)$.

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

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

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

206 **► Proposition 5 ([7]).** Let f be a space function over a cs $(\text{Con}, \sqsubseteq)$. Then (1) f is monotonic
 207 and (2) f preserves arbitrary joins.

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

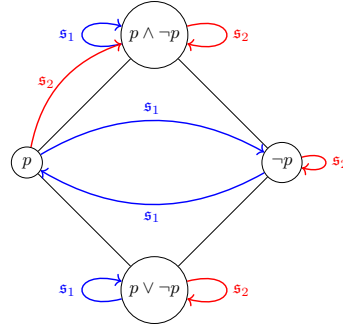
209 **► Definition 6 (Spatial Constraint Systems).** A spatial cs (scs) is a cs $\mathbf{C} = (\text{Con}, \sqsubseteq)$ equipped
 210 with a possibly infinite tuple $\mathfrak{s} = (\mathfrak{s}_i)_{i \in G}$ of space functions from $\mathcal{S}(\mathbf{C})$.

211 We shall use $(\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ to denote an scs with a tuple $(\mathfrak{s}_i)_{i \in G}$. We refer to G and
 212 \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
 213 agent i . Subsets of G are also referred to as groups of agents (or sub-groups of G).

214 Let us illustrate a simple scs.

215 **► Example 7.** The scs $(\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in \{1,2\}})$ in Fig.1 is given by the complete lattice \mathbf{M}_2 and
 216 two agents. We have $\text{Con} = \{p \vee \neg p, p, \neg p, p \wedge \neg p\}$ and $c \sqsubseteq d$ iff c is a logical consequence
 217 of d . The top element *false* is $p \wedge \neg p$, the bottom element *true* is $p \vee \neg p$, and p and $\neg p$
 218 are incomparable with each other. The set of agents is $\{1, 2\}$ with space functions \mathfrak{s}_1 and
 219 \mathfrak{s}_2 : For agent 1, $\mathfrak{s}_1(p) = \neg p$, $\mathfrak{s}_1(\neg p) = p$, $\mathfrak{s}_1(\text{false}) = \text{false}$, $\mathfrak{s}_1(\text{true}) = \text{true}$, and for agent 2,
 220 $\mathfrak{s}_2(p) = \text{false} = \mathfrak{s}_2(\text{false})$, $\mathfrak{s}_2(\neg p) = \neg p$, $\mathfrak{s}_2(\text{true}) = \text{true}$. The intuition is that the agent 2
 221 sees no difference between p and *false* while agent 1 interprets $\neg p$ as p and vice versa.

222 More involved examples of scs include meaningful families of structures from logic and
 223 economics such as modal algebras with continuous modal operators, Kripke structures and
 224 Aumann structures (see [15]). In Ex.19 we describe the Aumann structure example from
 225 [15]. We shall also 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 .

226 **4 Distributed Information**

227 In this section we characterize the notion of collective information of a group of agents. We
 228 begin with some intuition. Roughly speaking, the *distributed (or collective) information* of a
 229 group I is the join of each piece of information that resides in the space of *some* $i \in I$. The
 230 distributed information of I w.r.t. c is the distributive information of I that can be derived
 231 from c . We are interested in formalizing whether a given e can be derived from the collective
 232 information of the group I w.r.t. c .

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

235 ► **Example 8.** Consider a scs $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ where $G = \mathbb{N}$ and (Con, \sqsubseteq) is a constraint
 236 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 spatial constraint c specifies the situation
 237 where a , $a \rightarrow b$ and $b \rightarrow e$ are in the spaces of agent 1, 2 and 3, respectively. Neither agent
 238 holds e in their space in c . Nevertheless, the information e can be derived from the collective
 239 information of the three agents w.r.t. c , since from Prop.3 we have $a \sqcup (a \rightarrow b) \sqcup (b \rightarrow e) \sqsupseteq e$.
 240 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
 241 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
 242 (see Section 2), it may be the case that no agent $i \in \mathbb{N}$ holds e' in their space; e.g., if $e' \sqsupset a_i$
 243 for any $i \in \mathbb{N}$. Yet, from our assumption, e' can be derived from the collective information
 244 w.r.t. c' of all the agents in \mathbb{N} , i.e., $\bigsqcup_{i \in \mathbb{N}} a_i$.

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

248 ► **Definition 9 (Agent and Join Projections).** Let $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be a scs. Given $i \in G$,
 249 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
 250 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
 251 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 $\pi_I(c) \sqsupseteq e$.

252 The i -projection of an agent i of c naturally represents the join of all the information of
 253 agent i in c . The I -join projection of group I joins individual i -projections of c for $i \in I$. This
 254 projection can be used as a sound mechanism for reasoning about distributed-information:
 255 If e is I -join derivable from c then it follows from the distributed-information of I w.r.t. c .
 256 Consider the following example.

257 ► **Example 10.** Let c be as in Ex.8. We have $\pi_1(c) \sqsupseteq a$, $\pi_2(c) \sqsupseteq (a \rightarrow b)$, $\pi_3(c) \sqsupseteq (b \rightarrow e)$.
 258 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
 259 for c' as in Ex.8 we conclude that e' is I -join derivable from c' since $\pi_{\mathbb{N}}(c') = \bigsqcup_{i \in \mathbb{N}} \pi_i(c) \sqsupseteq$
 260 $\bigsqcup_{i \in \mathbb{N}} a_i \sqsupseteq e'$.

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

263 ► **Example 11.** 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
 264 disjunction of assertions: d specifies that b is present in the space of agent 1 or in the space
 265 of agent 2 though not exactly in which one. Thus from d we should be able to conclude that b
 266 belongs to the space of *some* agent in $\{1, 2\}$. Nevertheless, in general b is not I -join derivable
 267 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
 268 the scs in Fig.2a and take $b = \neg p$. We have $\pi_{\{1,2\}}(d) = \pi_1(d) \sqcup \pi_2(d) = \text{true} \sqcup \text{true} = \text{true} \not\sqsupseteq b$.
 269 One can generalize the example to infinitely many agents: Consider the scs in Ex.8. Let
 270 $d' \stackrel{\text{def}}{=} \prod_{i \in \mathbb{N}} \mathfrak{s}_i(b')$. We should be able to conclude from d' that b' is in the space of *some* agent
 271 in \mathbb{N} but, in general, b' is not \mathbb{N} -join derivable from d' .

272 4.1 Distributed Spaces

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

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

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

284 ► **Definition 12 (Space Functions Order).** Let $\mathbf{C} = (\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be a spatial cs. Given
 285 $f, g \in \mathcal{S}(\mathbf{C})$, define $f \sqsubseteq_s g$ iff $f(c) \sqsubseteq g(c)$ for every $c \in \text{Con}$. We shall use \mathbf{C}_s to denote the
 286 partial order $(\mathcal{S}(\mathbf{C}), \sqsubseteq_s)$; the set of all space functions ordered by \sqsubseteq_s .

287 A very important fact for the design of our structure is that the set of space functions
 288 $\mathcal{S}(\mathbf{C})$ can be made into a complete lattice.

289 ► **Lemma 13.** Let $\mathbf{C} = (\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be a spatial cs. Then \mathbf{C}_s is a complete lattice.

290 4.2 Distributed Spaces as Maximum Spaces.

291 Let us consider the lattice of space functions $\mathbf{C}_s = (\mathcal{S}(\mathbf{C}), \sqsubseteq_s)$. Suppose that f and g are
 292 space functions in \mathbf{C}_s with $f \sqsubseteq_s g$. Intuitively, every piece of information c in the space
 293 represented by g is also in the space represented by f since $f(c) \sqsubseteq g(c)$ for every $c \in \text{Con}$.
 294 This can be interpreted as saying that the space represented by g is included in the space
 295 represented by f ; in other words the bigger the space, the smaller the function that represents
 296 it in the lattice \mathbf{C}_s .

297 Following the above intuition, the order relation \sqsubseteq_s of \mathbf{C}_s represents (reverse) space
 298 inclusion and the join and meet operations in \mathbf{C}_s represent intersection and union of spaces.

8 Reasoning about Distributed Knowledge

299 The biggest and the smallest spaces are represented by the bottom and the top elements of
 300 the lattice \mathbf{C}_s , here called λ_{\perp} and λ_{\top} and defined as follows.

301 ► **Definition 14** (Top and Bottom Spaces). *For every $c \in \text{Con}$, define $\lambda_{\perp}(c) \stackrel{\text{def}}{=} \text{true}$,
 302 $\lambda_{\top}(c) \stackrel{\text{def}}{=} \text{true}$ if $c = \text{true}$ and $\lambda_{\top}(c) \stackrel{\text{def}}{=} \text{false}$ if $c \neq \text{true}$.*

303 The distributed space Δ_I of a group I can be viewed as the function that represents the
 304 smallest space that includes all the local information of the agents in I . From the above
 305 intuition, Δ_I should be the *greatest space function* below the space functions of the agents in
 306 I . The existence of such a function follows from completeness of $(\mathcal{S}(\mathbf{C}), \sqsubseteq_s)$ (Lemma 13).

307 ► **Definition 15** (Distributed Space Functions). *Let \mathbf{C} be a scs $(\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. The dis-*
 308 *tributed spaces of \mathbf{C} is given by $\Delta = (\Delta_I)_{I \subseteq G}$ where*

$$309 \quad \Delta_I \stackrel{\text{def}}{=} \max\{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq_s \mathfrak{s}_i \text{ for every } i \in I\}.$$

310 *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*
 311 *Δ_I as the (distributed) space of the group I .*

312 It follows from Lemma 13 that $\Delta_I = \sqcap\{\mathfrak{s}_i \mid i \in I\}$ (where \sqcap is the meet in the complete
 313 lattice $(\mathcal{S}(\mathbf{C}), \sqsubseteq_s)$). Fig.2b illustrates a scs and its distributed space $\Delta_{\{1,2\}}$.

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

317 ► **Theorem 16.** *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a scs $(\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that*
 318 *$K, J \subseteq I \subseteq G$. (1) $\Delta_I = \lambda_{\top}$ if $I = \emptyset$, (2) $\Delta_I = \mathfrak{s}_i$ if $I = \{i\}$, (3) $\Delta_J(a) \sqcup \Delta_K(b) \sqsupseteq \Delta_I(a \sqcup b)$,*
 319 *and (4) $\Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \sqsupseteq \Delta_I(c)$ if $(\text{Con}, \sqsubseteq)$ is a constraint frame.*

320 Recall that λ_{\top} corresponds to the empty space (see Def.14). The first property realizes the
 321 intuition that the empty subgroup \emptyset *does not* have any information whatsoever distributed
 322 w.r.t. a consistent c : for if $c \sqsupseteq \Delta_{\emptyset}(e)$ and $c \neq \text{false}$ then $e = \text{true}$. Intuitively, the second
 323 property says that the function Δ_I for the group of one agent must be the agent's space
 324 function. The third property states that a group can join the information of its subgroups.
 325 The last property uses constraint implication, hence the constraint frame condition, to express
 326 that by joining the information a and $a \rightarrow c$ of their subgroups, the group I can obtain c .

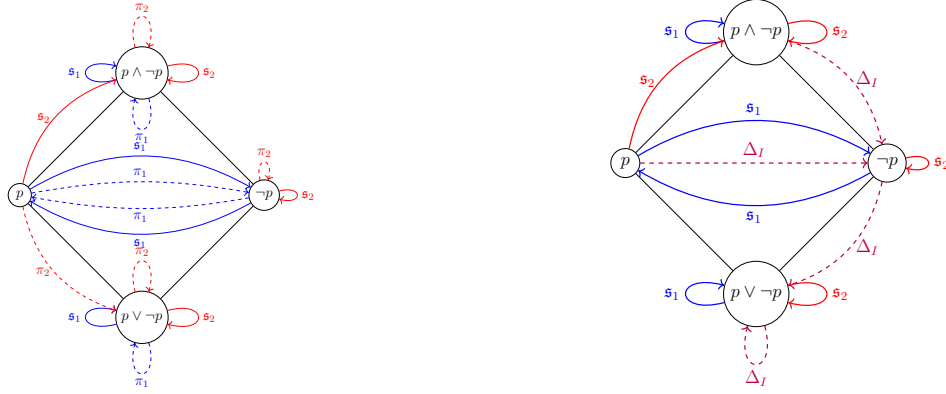
327 Let us illustrate how to derive information of a group from smaller ones using Thm.16.

328 ► **Example 17.** Let $c = \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \rightarrow b) \sqcup \mathfrak{s}_3(b \rightarrow e)$ as in Ex.8. We want to prove that e
 329 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
 330 in Thm.16 we obtain $c \sqsupseteq \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \rightarrow b) = \Delta_{\{1\}}(a) \sqcup \Delta_{\{2\}}(a \rightarrow b) \sqsupseteq \Delta_{\{1,2\}}(b)$, and then
 331 $c \sqsupseteq \Delta_{\{1,2\}}(b) \sqcup \mathfrak{s}_3(b \rightarrow e) = \Delta_{\{1,2\}}(b) \sqcup \Delta_{\{3\}}(b \rightarrow e) \sqsupseteq \Delta_{\{1,2,3\}}(e)$ as wanted.

332 ► **Remark 18** (Continuity). The example with infinitely many agents in Ex.8 illustrates well
 333 why we require our spaces to be continuous in the presence of possibly infinite groups. Clearly
 334 $c' = \bigsqcup_{i \in \mathbb{N}} \mathfrak{s}_i(a_i) \sqsupseteq \bigsqcup_{i \in \mathbb{N}} \Delta_{\mathbb{N}}(a_i)$. By continuity, $\bigsqcup_{i \in \mathbb{N}} \Delta_{\mathbb{N}}(a_i) = \Delta_{\mathbb{N}}(\bigsqcup_{i \in \mathbb{N}} a_i)$ which indeed
 335 captures the idea that each a_i is in the distributed space $\Delta_{\mathbb{N}}$.

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

338 We conclude this subsection with an important family of scs's from mathematical eco-
 339 nomics: Aumann structures. We illustrate that the notion of distributed knowledge in these
 340 structures is an instance of a distributed space.



(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 .

341 ▶ **Example 19.** *Aumann Constraint Systems.* Aumann structures [13] are an *event-based*
 342 approach to modelling knowledge. An Aumann structure is a tuple $\mathcal{A} = (S, \mathcal{P}_1, \dots, \mathcal{P}_n)$
 343 where S is a set of states and each \mathcal{P}_i is a partition on S for agent i . The partitions are
 344 called *information sets*. If two states t and u are in the same information set for agent
 345 i , it means that in state t agent i considers state u possible, and vice versa. An *event* in
 346 an Aumann structure is any subset of S . Event e holds at state t if $t \in e$. The set $\mathcal{P}_i(s)$
 347 denotes the information set of \mathcal{P}_i containing s . The event of *agent i knowing e* is defined as
 348 $K_i(e) = \{s \in S \mid \mathcal{P}_i(s) \subseteq e\}$, and the *distributed knowledge of an event e among the agents*
 349 *in a group I* is defined as $D_I(e) = \{s \in S \mid \bigcap_{i \in I} \mathcal{P}_i(s) \subseteq e\}$.

350 An Aumann structure can be seen as a spatial constraint system $\mathbf{C}(\mathcal{A})$ with events as
 351 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
 352 $e_2 \subseteq e_1$. The operators join (\sqcup) and meet (\sqcap) are intersection (\cap) and union (\cup) of events,
 353 respectively; $true = S$ and $false = \emptyset$. The space functions are the knowledge operators,
 354 i.e., $\mathfrak{s}_i(c) = K_i(c)$. From these definitions and since meets are unions one can easily verify
 355 that $\Delta_I(c) = D_I(c)$ which shows the correspondence between distributed information and
 356 distributed knowledge.

357 4.3 Distributed Spaces as Group Distributions Candidates.

358 We now wish to single out a few fundamental properties on tuples of self-maps that can be
 359 used to characterize distributed spaces.

360 ▶ **Definition 20** (Distribution Candidates). *Let \mathbf{C} be a scs $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. A tuple $\delta =$
 361 $(\delta_I)_{I \subseteq G}$ of self-maps on Con is a group distribution candidate (gdc) of \mathbf{C} if for each $I, J \subseteq G$:*
 362 *(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 $I \subseteq J$.*

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

371 Properties D1-D3, however, do not determine Δ uniquely. In fact, there could be infinitely-
 372 many tuples of space functions that satisfy them. For example, if we were to chose $\delta_\emptyset = \lambda_\top$,
 373 $\delta_{\{i\}} = \mathfrak{s}_i$ for every $i \in G$, and $\delta_I = \lambda_\perp$ whenever $|I| > 1$ then D1, D2 and D3 would
 374 be trivially met. But these space functions would not capture our intended meaning of
 375 distributed spaces: E.g., we would have $true \sqsupseteq \delta_I(e)$ for every e thus implying that any e
 376 could be distributed in the empty information $true$ amongst the agents in $I \neq \emptyset$.

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

379 ► **Theorem 21** (Max gcd). *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$.
 380 Then $(\Delta_I)_{I \subseteq G}$ is a gcd of \mathbf{C} and if $(\delta_I)_{I \subseteq G}$ is a gcd of \mathbf{C} then $\delta_I \sqsubseteq_s \Delta_I$ for each $I \subseteq G$.*

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

382 ► **Example 22.** Let $c = \mathfrak{s}_1(a) \sqcup \mathfrak{s}_2(a \rightarrow b) \sqcup \mathfrak{s}_3(b \rightarrow e)$ as in Ex.8. We want to prove $c \sqsupseteq \Delta_I(e)$
 383 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
 384 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
 385 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
 386 counter-example in Ex.11 with $d = \mathfrak{s}_1(b) \sqcap \mathfrak{s}_2(b)$. We wish to prove $d \sqsupseteq \Delta_I(b)$ for $I = \{1, 2\}$.
 387 I.e., that b can be derived from d as being in a space of a member of $\{1, 2\}$. Using D.1 and
 388 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.

389 The characterization of distributed spaces by Thm.21 provide us with a convenient proof
 390 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
 391 tuple is a gcd and that $f_I \sqsupseteq_s \Delta_I$ for all $I \subseteq G$. We use this mechanism in Section 5.

392 4.4 Group Projections

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

395 ► **Definition 23** (Group Projection). *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs $\mathbf{C} =$
 396 $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Given the set $I \subseteq G$, the I -group projection of $c \in Con$ is defined as
 397 $\Pi_I(c) \stackrel{\text{def}}{=} \bigsqcup \{e \mid c \sqsupseteq \Delta_I(e)\}$. We say that e is I -group derivable from c iff $\Pi_I(c) \sqsupseteq e$.*

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

400 ► **Proposition 24.** *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. For every
 401 $c, e \in Con$, (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) $\Pi_I(c) \sqsupseteq \pi_I(c)$.*

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

407 4.5 Group Compactness.

408 Suppose that an *infinite* group of agents I can derive e from c (i.e., $c \sqsupseteq \Delta_I(e)$). A legitimate
 409 question is whether there exists a *finite* sub-group J of agents from I that can also derive e
 410 from c . The following theorem provides a positive answer to this question provided that e is
 411 a compact element (see Section 2) and I -join derivable from c .

412 ► **Theorem 25** (Group Compactness). *Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs*
 413 $\mathbf{C} = (\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. *Suppose that $c \sqsupseteq \Delta_I(e)$. If e is compact and I -join derivable from c*
 414 *then there exists a finite set $J \subseteq I$ such that $c \sqsupseteq \Delta_J(e)$.*

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

416 ► **Example 26.** Consider the example with infinitely many agents in Ex.8. We have
 417 $c' = \bigsqcup_{i \in \mathbb{N}} \mathfrak{s}_i(a_i)$ for some increasing chain $a_0 \sqsubseteq a_1 \sqsubseteq \dots$ and e' s.t. $e' \sqsubseteq \bigsqcup_{i \in \mathbb{N}} a_i$. Notice
 418 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
 419 Thm.25 there must be a finite subset $J \subseteq \mathbb{N}$ such that $c' \sqsupseteq \Delta_J(e')$.

420 5 Computing Distributed Information

421 Let us consider a *finite* scs $\mathbf{C} = (\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ with distributed spaces $(\Delta_I)_{I \subseteq G}$. By finite
 422 scs we mean that Con and G are finite sets. Let us consider the problem of computing Δ_I :
 423 Given a set $\{\mathfrak{s}_i\}_{i \in I}$ of space functions, we wish to find the greatest space function f such
 424 that $f \sqsubseteq \mathfrak{s}_i$ for all $i \in I$ (see Def.15).

425 Because of the finiteness assumption, the above problem can be rephrased in simpler
 426 terms: *Given a finite lattice L and a finite set S of join-homomorphisms on L , find the*
 427 *greatest join-homomorphism below all the elements of S .* Even in small lattices with four
 428 elements and two space functions, finding such greatest function may not be immediate, e.g.,
 429 for $S = \{\mathfrak{s}_1, \mathfrak{s}_2\}$ and the lattice in Fig.1 the answer is given Fig.2b.

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

$$432 \max\{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq \mathfrak{s}_i \text{ for all } i \in I\} = \bigsqcup \{f \in \mathcal{S}(\mathbf{C}) \mid f \sqsubseteq \mathfrak{s}_i \text{ for all } i \in I\} = \bigsqcap \{\mathfrak{s}_i \mid i \in I\}$$

433 A *naive (meet-based) approach* would be to compute $\Delta_I(c)$ by taking the point-wise meet
 434 construction $\sigma_I(c) \stackrel{\text{def}}{=} \bigsqcap \{\mathfrak{s}_i(c) \mid i \in I\}$ for each $c \in \text{Con}$. But this does not work in general
 435 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
 436 $\sigma_I \sqsupseteq_s \Delta_I$ but σ_I may not even be a space function as shown in Fig.3a.

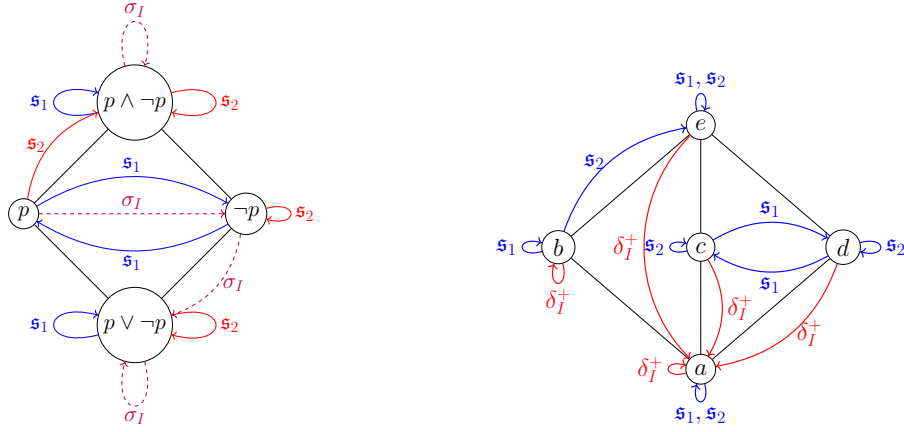
437 A *brute force (join-based) solution* to computing $\Delta_I(c)$ can be obtained by generating the
 438 set $\{f(c) \mid f \in \mathcal{S}(\mathbf{C}) \text{ and } f \sqsubseteq \mathfrak{s}_i \text{ for all } i \in I\}$ and taking its join. This approach works since
 439 the join of a set of space functions S can be computed point-wise: $(\bigsqcup S)(c) = \bigsqcup \{f(c) \mid f \in S\}$.
 440 However, the number of such functions in $\mathcal{S}(\mathbf{C})$ can be at least factorial in the size of Con .
 441 For constraint frames, which under the finite assumption coincides with distributive lattices,
 442 the size of $\mathcal{S}(\mathbf{C})$ can be non-polynomial in the size of Con .

443 ► **Proposition 27** (Lower Bounds on Number of Space Functions). *For every $n \geq 2$, there*
 444 *exists a cs $\mathbf{C} = (\text{Con}, \sqsubseteq)$ such that $|\mathcal{S}(\mathbf{C})| \geq (n-2)!$ and $n = |\text{Con}|$. For every $n \geq 1$, there*
 445 *exists a constraint frame $\mathbf{C} = (\text{Con}, \sqsubseteq)$ such that $|\mathcal{S}(\mathbf{C})| \geq n^{\log_2 n}$ and $n = |\text{Con}|$.*

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

451 5.1 Distributed Spaces in Distributed Lattices

452 Here we shall illustrate some pleasant compositionality properties of distributed spaces that
 453 can be used for computing Δ_I in distributed lattices (constraint frames). These properties



(a) For $I = \{1, 2\}$, $\sigma_I(c) = \prod_{i \in I} s_i(c)$ 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 28) 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 M_2 (a) and the non-distributive lattice M_3 (b).

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

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

458 ► **Lemma 28.** Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $\mathbf{C} = (Con, \sqsubseteq, (s_i)_{i \in G})$.
 459 Suppose that (Con, \sqsubseteq) is a constraint frame. Let $\delta_I^+ : Con \rightarrow Con$, with $I \subseteq G$, be the
 460 function $\delta_I^+(c) \stackrel{\text{def}}{=} \prod \{ \sqcup_{i \in I} s_i(a_i) \mid (a_i)_{i \in I} \in Con^I \text{ and } \sqcup_{i \in I} a_i \sqsupseteq c \}$. Then $\Delta_I = \delta_I^+$.

461 The above lemma basically says that $\Delta_I(c)$ is the greatest information below all possible
 462 combinations of information in the spaces of the agents in I that derive c . The proof that
 463 $\delta_I^+ \sqsupseteq_s \Delta_I$ uses the fact that space functions preserve joins. The proof that $\delta_I^+ \sqsubseteq_s \Delta_I$ proceeds
 464 by showing that $(\delta_I^+)_{I \subseteq G}$ is a group distribution candidate (Def.20). Distributivity of the
 465 lattice (Con, \sqsubseteq) is crucial for this direction. In fact without it $\Delta_I = \delta_I^+$ does not necessarily
 466 hold as shown by the following counter-example.

467 ► **Example 29.** Consider the non-distributive lattice M_3 and the space functions s_1 and
 468 s_2 in Figure 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,
 469 $\delta_I^+(b \sqcup c) \neq \delta_I^+(b) \sqcup \delta_I^+(c)$, i.e., δ_I^+ is not a space function.

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

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

476 1. $\Delta_I(c) = \prod \{ \Delta_J(a) \sqcup \Delta_K(b) \mid a, b \in Con \text{ and } a \sqcup b \sqsupseteq c \}$. (1)

477 2. $\Delta_I(c) = \prod \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \mid a \in Con \}$. (2)

478 3. $\Delta_I(c) = \prod \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \mid a \in Con \text{ and } a \sqsubseteq c \}$. (3)

479 The above properties bear witness to the inherent compositional nature of our notion of
 480 distributed space. This nature will be exploited by the algorithms below. The first property
 481 in Thm.30 essentially reformulates Lemma 28 in terms of subgroups rather than agents. It
 482 can be proven by replacing $\Delta_J(a)$ and $\Delta_K(b)$ by $\delta_J^+(a)$ and $\delta_K^+(b)$, defined in Lemma 28 and
 483 using distributivity of joins over meets. The second and third properties in Theorem 30 are
 484 pleasant simplifications of the first using heyting implication. These properties realize the
 485 intuition that by joining the information a and $a \rightarrow c$ of their subgroups, the group I can
 486 obtain c .

487 5.2 Algorithms for Distributed Lattices

488 Recall that λ_\top represents the empty distributed space (see Def.14). Given finite scs $\mathbf{C} =$
 489 $(Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ with distributed spaces $(\Delta_I)_{I \subseteq G}$, the recursive function DELTAPART3(I, c)
 490 in Algorithm 1 computes $\Delta_I(c)$ for any given $c \in Con$. Its correctness, assuming that (Con, \sqsubseteq)
 491 is a constraint frame (i.e., a distributed lattice), follows from Thm.30(3). Termination follows
 492 from the finiteness of \mathbf{C} and the fact the sets J and K in the recursive calls form a partition
 493 of I . Notice that we select a partition (in halves) rather than any two sets K, J satisfying
 494 the condition $I = J \cup K$ to avoid significant recalculation.

Algorithm 1 Function DELTAPART3(I, c) computes $\Delta_I(c)$

```

1: function DELTAPART3( $I, c$ ) ▷ Computes  $\Delta_I(c)$ 
2:   if  $I = \emptyset$  then
3:     return  $\lambda_\top(c)$ 
4:   else if  $I = \{i\}$  then
5:     return  $\mathfrak{s}_i(c)$ 
6:   else
7:      $\{J, K\} \leftarrow \text{PARTITION}(I)$  ▷ returns a partition  $\{J, K\}$  of  $I$  s.t.,  $|J| = \lfloor |I|/2 \rfloor$ 
8:     return  $\prod \{\text{DELTAPART3}(J, a) \sqcup \text{DELTAPART3}(K, a \rightarrow c) \mid a \in Con \text{ and } a \sqsubseteq c\}$ .
```

495 *Algorithms.* Notice DELTAPART3(I, c) computes $\Delta_I(c)$ using Thm.30(3). By modifying
 496 Line 8 with the corresponding meet operations, we obtain two variants of DELTAPART3
 497 that use, instead of Thm.30(3), the Properties Thm.30(1) and Thm.30(2). We call them
 498 DELTAPART1 and DELTAPART2. Finally, we also obtain a non-recursive algorithm that
 499 outputs $\Delta_I(c)$ by computing $\delta_I^+(c)$ in Lemma 28 in the obvious way: Computing the meet
 500 of elements of the form $\prod_{i \in I} \mathfrak{s}_i(a_i)$ for every tuple $(a_i)_{i \in I}$ such that $\prod_{i \in I} a_i \sqsupseteq c$. We call it
 501 DELTA+.

502 *Worst-case time complexity.* We assume that binary distributive lattice operations $\prod, \sqcup,$
 503 and \rightarrow are computed in $O(1)$ time. We also assume a fixed group I of size $m = |I|$ and express
 504 the time complexity for computing Δ_I in terms of $n = |Con|$, the size of the set of constraints.
 505 The above-mentioned algorithms compute the value $\Delta_I(c)$. The worst-case time complexity
 506 for computing the function Δ_I is in (1) $O(mn^{1+m})$ using DELTA+, (2) $O(mn^{1+2 \log_2 m})$ using
 507 DELTAPART1, and (3) $O(mn^{1+\log_2 m})$ using DELTAPART2 and DELTAPART3.

508 5.3 Algorithm for Arbitrary Lattices

509 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
 510 space function Δ_I under a collection $\{\mathfrak{s}_i\}_{i \in I}$ can be computed by successive approximations,
 511 starting with some (not necessarily space) function known to be less than all $\{\mathfrak{s}_i\}_{i \in I}$. Assume
 512 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

513 point is $\sigma(u) = \sqcap\{\mathfrak{s}_i(u) \mid i \in I\}$, for all $u \in Con$. By definition of \sqcap , $\sigma(u)$ is the biggest
 514 function under all functions in $\{\mathfrak{s}_i\}_{i \in I}$, hence $\sigma \sqsupseteq \Delta_I$. The algorithm computes decreasing
 515 upper bounds of Δ_I by correcting σ values not conforming to the space function property
 516 $\sigma(u) \sqcup \sigma(v) = \sigma(u \sqcup v)$. The correction decreases σ and maintains the invariant $\sigma \sqsupseteq \Delta_I$.

517 There are two ways of correcting σ values: (1) when $\sigma(u) \sqcup \sigma(v) \sqsubset \sigma(u \sqcup v)$, assign
 518 $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$ and (2) when $\sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(u \sqcup v)$, assign $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$
 519 and also $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$. It can be shown that the assignments in both cases should
 520 decrease σ while preserving the $\sigma \sqsupseteq \Delta_I$ invariant.

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

Algorithm 2 DELTAGEN finds Δ_I

```

 $\sigma(u) \leftarrow \sqcap\{\mathfrak{s}_i(u) \mid i \in I\}$  ▷ for all  $u \in Con$ 
while  $u, v \in Con \wedge \sigma(u) \sqcup \sigma(v) \neq \sigma(u \sqcup v)$  do
  if  $\sigma(u) \sqcup \sigma(v) \sqsubset \sigma(u \sqcup v)$  then ▷ case (1)
     $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$ 
  else ▷ case (2)
     $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$ 
     $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$ 

```

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

529 6 Conclusions and Related Work

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

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

541 Our work is inspired by the epistemic concept of distributed knowledge [5]. Knowledge
 542 in distributed systems was discussed in [11], based on interpreting distributed systems using
 543 Hintikka's notion of possible worlds. In this definition of distributed knowledge, the system
 544 designer ascribes knowledge to processors (agents) in each global state (a processor's local
 545 state). In [12] the authors present a general framework to formalize the knowledge of a
 546 group of agents, in particular the notion of distributed knowledge. The authors consider
 547 distributed knowledge as knowledge that is distributed among the agents belonging to a
 548 given group, without any individual agent necessarily having this knowledge. In [13] the

549 authors study knowledge and common knowledge in situations with infinitely many agents.
 550 The authors highlight the importance of reasoning about infinitely many agents in situations
 551 where the number of agents is not known in advance. Their work does not address distributed
 552 knowledge but points out potential technical difficulties in their future work.

 553 ——— **References** ———

- 554 **1** Samson Abramsky and Achim Jung. Domain theory. In Samson Abramsky et al., editors,
 555 *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Oxford University Press, 1994.
- 556 **2** Frank S. Boer, Alessandra Di Pierro, and Catuscia Palamidessi. Nondeterminism and infinite
 557 computations in constraint programming. *Theoretical Computer Science*, pages 37–78, 1995.
- 558 **3** Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university
 559 press, 2nd edition, 2002.
- 560 **4** Joan-María Esteban and Debraj Ray. On the measurement of polarization. *Econometrica*,
 561 62(4):819–851, 1994.
- 562 **5** Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about*
 563 *knowledge*. MIT press Cambridge, 4th edition, 1995.
- 564 **6** Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael Mislove,
 565 and Dana S. Scott. *Continuous lattices and domains*. Cambridge University Press, 2003.
- 566 **7** Michell Guzman, Stefan Haar, Salim Perchy, Camilo Rueda, and Frank Valencia. Belief,
 567 knowledge, lies and other utterances in an algebra for space and extrusion. *Journal of Logical*
 568 *and Algebraic Methods in Programming*, 86(1):107–133, 2017.
- 569 **8** Michell Guzman, Salim Perchy, Camilo Rueda, and Frank Valencia. Deriving Inverse Operators
 570 for Modal Logic. In *Theoretical Aspects of Computing – ICTAC 2016*, volume 9965 of *Theoretical*
 571 *Aspects of Computing – ICTAC 2016*, pages 214–232. Springer, 2016.
- 572 **9** Michell Guzmán, Salim Perchy, Camilo Rueda, and Frank Valencia. Characterizing Right
 573 Inverses for Spatial Constraint Systems with Applications to Modal Logic. *Theoretical*
 574 *Computer Science*, 744(56–77), October 2018. URL: <https://hal.inria.fr/hal-01675010>.
- 575 **10** Stefan Haar, Salim Perchy, Camilo Rueda, and Frank Valencia. An algebraic view of space/belief
 576 and extrusion/utterance for concurrency/epistemic logic. In *PPDP 2015*, pages 161–172. ACM,
 577 2015.
- 578 **11** Joseph Y Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual*
 579 *review of computer science*, 2(1):37–68, 1987.
- 580 **12** Joseph Y Halpern and Yoram Moses. Knowledge and common knowledge in a distributed
 581 environment. *Journal of the ACM (JACM)*, 37(3):549–587, 1990.
- 582 **13** Joseph Y Halpern and Richard A Shore. Reasoning about common knowledge with infinitely
 583 many agents. *Information and Computation*, 191(1):1–40, 2004.
- 584 **14** Werner Hildenbrand. On economies with many agents. *Journal of Economic Theory*, 2(2):161
 585 – 188, 1970.
- 586 **15** Sophia Knight, Catuscia Palamidessi, Prakash Panangaden, and Frank Valencia. Spatial and
 587 epistemic modalities in constraint-based process calculi. In *CONCUR 2012*, volume 7454 of
 588 *LNCS*, pages 317–332. Springer, 2012.
- 589 **16** Sophia Knight, Prakash Panangaden, and Frank Valencia. Computing with epistemic and
 590 spatial modalities. Technical report available upon request, 2019.
- 591 **17** Camilo Rueda and Frank Valencia. On validity in modelization of musical problems by ccp.
 592 *Soft Computing*, 8(9):641–648, 2004.
- 593 **18** Vijay A Saraswat, Martin Rinard, and Prakash Panangaden. Semantic foundations of concur-
 594 rent constraint programming. In *POPL ’91*, pages 333–352. ACM, 1991.

595 **A Proofs**596 **Proof of Lemma 13.**

597 Let $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$ be a spatial cs. Then \mathbf{C}_s is a complete lattice.

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

600 Let $S = \{\mathfrak{s}_i\}_{i \in I}$ be a family of space functions with arbitrary indexing set $I \subseteq G$. For
601 every constraint $c \in Con$, we let $(\sqcup S)(c) = \sqcup_{i \in I} \mathfrak{s}_i(c)$. We shall show that $\sqcup S$ is a space
602 function.

603 ■ $(\sqcup S)(true) = \sqcup_{i \in I} \mathfrak{s}_i(true) = \sqcup true = true.$

604 ■ $(\sqcup S)(c \sqcup d) = (\sqcup S)(c) \sqcup (\sqcup S)(d).$

$$\begin{aligned}
 605 \quad (\sqcup S)(c \sqcup d) &= \sqcup_{i \in I} \mathfrak{s}_i(c \sqcup d) \\
 606 &= \sqcup_{i \in I} (\mathfrak{s}_i(c) \sqcup \mathfrak{s}_i(d)) \\
 607 &= (\sqcup_{i \in I} \mathfrak{s}_i(c)) \sqcup (\sqcup_{i \in I} \mathfrak{s}_i(d)) \\
 608 &= (\sqcup S)(c) \sqcup (\sqcup S)(d) \\
 609
 \end{aligned}$$

610 ■ Continuity: $(\sqcup S)(\sqcup D) = \sqcup_{d \in D} (\sqcup S)(d)$ for any directed set D .

611 Suppose that D is a directed set. From definition $(\sqcup S)(\sqcup D) = \sqcup_{i \in I} \mathfrak{s}_i(\sqcup D)$. By the con-
612 tinuity of each space function \mathfrak{s}_i , $\sqcup_{i \in I} \mathfrak{s}_i(\sqcup D) = \sqcup_{i \in I} \sqcup_{d \in D} \mathfrak{s}_i(d) = \sqcup_{d \in D} \sqcup_{i \in I} \mathfrak{s}_i(d) =$
613 $\sqcup_{d \in D} (\sqcup S)(d)$, as required.

614 ◀

615 **Proof of Theorem 16.**

616 Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that
617 $I \subseteq G$.

- 618 1. $\Delta_I = \lambda_{\top}$ if $I = \emptyset$,
- 619 2. $\Delta_I = \mathfrak{s}_i$ if $I = \{i\}$,
- 620 3. $\Delta_J(a) \sqcup \Delta_K(b) \sqsupseteq \Delta_I(a \sqcup b)$ if $K, J \subseteq I$,
- 621 4. $\Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \sqsupseteq \Delta_I(c)$ if $K, J \subseteq I$ and (Con, \sqsubseteq) is a constraint frame.

622 **Proof.** The proof of part (1) follows from definition of λ_{\top} (see Definition 14) and the fact that
623 $max \emptyset = true$. Part (2) is immediate from Definition 15 of Δ_I . For property (3), recall that
624 the bigger the group the smaller the space function associated to it. Thus, if $K, J \subseteq I$ note
625 that $\Delta_J(a) \sqsupseteq \Delta_I(a)$ and $\Delta_K(b) \sqsupseteq \Delta_I(b)$, then $\Delta_J(a) \sqcup \Delta_K(b) \sqsupseteq \Delta_I(a) \sqcup \Delta_I(b) = \Delta_I(a \sqcup b)$.
626 To prove (4), we use part (3) with $a = a$ and $b = a \rightarrow c$, and Proposition 3. ◀

627 **Proof of Theorem 21.**

628 Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Then (1) $(\Delta_I)_{I \subseteq G}$
629 is a gcd of \mathbf{C} and (2) if $(\delta_I)_{I \subseteq G}$ is a gcd of \mathbf{C} then $\delta_I \sqsubseteq_s \Delta_I$ for each $I \subseteq G$.

630 **Proof.** To prove (1) consider the properties (D.1)-(D.3) in Definition 20 for Δ_I . Property
631 (D.1) follows from definition of Δ_I (Definition 15). Property (D.2) is proven in Theorem 16
632 part (2). Property (D.3) is a consequence of: the bigger the group the smaller the space
633 function associated to it. To prove (2) note that δ_I is a space function and Δ_I is the maximum
634 of the space functions below \mathfrak{s}_i for every $i \in I$. ◀

Proof of Proposition 24.

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

Proof. For (1), firstly assume that $c \sqsupseteq \Delta_I(e)$. From Definition 23, $\Pi_I(c) = \bigsqcup \{d \mid c \sqsupseteq \Delta_I(d)\}$. Then, $\Pi_I(c) \sqsupseteq e$. Secondly, assume $\Pi_I(c) \sqsupseteq e$ and let $S = \{d \mid c \sqsupseteq \Delta_I(d)\}$. Note that $c \sqsupseteq \bigsqcup \{\Delta_I(d) \mid d \in S\}$. From continuity of Δ_I , we know that $\bigsqcup \{\Delta_I(d) \mid d \in S\} = \Delta_I(\bigsqcup S)$ and by monotonicity $c \sqsupseteq \Delta_I(\bigsqcup S) = \Delta_I(\Pi_I(c)) \sqsupseteq \Delta_I(e)$.

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

Finally, to prove property (3), by part (2) it is true that for every $\{i\} \subseteq I$, $\Pi_I \sqsupseteq_s \Pi_{\{i\}}$. Therefore, $\Pi_I \sqsupseteq_s \bigsqcup_{i \in I} \Pi_{\{i\}}$. Now, for every $c \in Con$, $\bigsqcup_{i \in I} \Pi_{\{i\}}(c) = \bigsqcup_{i \in I} \{\bigsqcup \{d \mid c \sqsupseteq \Delta_{\{i\}}(d)\}\} = \bigsqcup_{i \in I} \{\bigsqcup \{d \mid c \sqsupseteq \mathfrak{s}_i(d)\}\} = \bigsqcup_{i \in I} \{\pi_i(c)\} = \pi_I(c)$. Therefore, $\Pi_I(c) \sqsupseteq \pi_I(c)$, for every $c \in Con$. ◀

Proof of Theorem 25.

Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of an scs $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. 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 $c \sqsupseteq \Delta_J(e)$.

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

Define $S_I = \{\pi_J(c) \mid J \subseteq I \text{ and } J \text{ is finite}\}$. 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 $\bigsqcup S_I \sqsupseteq \bigsqcup S \sqsupseteq e$. But e is compact and S_I directed hence there must be $\pi_J(c) \in S_I$, with J a finite set, such that $\pi_J(c) \sqsupseteq e$. From Prop.24 (3) and Prop.24 (1), we conclude $c \sqsupseteq \Delta_J(e)$ as wanted. ◀

Proof of Proposition 27.

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

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

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) $f(T) = \bigsqcup_{t \in T} f(\{t\})$ 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 $\mathbf{C} = (Con, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that (Con, \sqsubseteq) is a constraint frame. Let $\delta_I^+ : Con \rightarrow Con$, with $I \subseteq G$, be the function

$$\delta_I^+(c) \stackrel{\text{def}}{=} \bigsqcap_{i \in I} \left\{ \bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid (a_i)_{i \in I} \in Con^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \right\}$$

for every $c \in Con$. Then $\Delta_I = \delta_I^+$.

678 **Proof.** Firstly, we are going to show that δ_I^+ is a group distribution candidate.

679 (D.1) δ_I^+ is a space function.

680 1. $\delta_I^+(true) = true$.

681 Note that $true \in \{\bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq true\}$, therefore $\delta_I^+(true) = true$.

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

683 We prove that δ_I^+ is monotonic: If $c \sqsupseteq d$ then $\delta_I^+(c) \sqsupseteq \delta_I^+(d)$. Assume $c \sqsupseteq d$. If $\bigsqcup_{i \in I} a_i \sqsupseteq c$,
684 then $\bigsqcup_{i \in I} a_i \sqsupseteq d$. Therefore, $\{\bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq c\} \subseteq \{\bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq d\}$
685 which implies $\delta_I^+(c) \sqsupseteq \delta_I^+(d)$.

686 Since δ_I^+ is monotonic, $\delta_I^+(c \sqcup d) \sqsupseteq \delta_I^+(c)$ and $\delta_I^+(c \sqcup d) \sqsupseteq \delta_I^+(d)$, thus $\delta_I^+(c \sqcup d) \sqsupseteq$
687 $\delta_I^+(c) \sqcup \delta_I^+(d)$.

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

$$\begin{aligned}
 & \delta_I^+(c) \sqcup \delta_I^+(d) \\
 &= \langle \text{Definition of } \delta_I^+(d) \rangle \\
 & \delta_I^+(c) \sqcup \prod \left\{ \bigsqcup_{i \in I} \mathfrak{s}_i(b_i) \mid \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 &= \langle \sqcup \text{ distributes over } \prod \rangle \\
 & \prod \left\{ \delta_I^+(c) \sqcup \bigsqcup_{i \in I} \mathfrak{s}_i(b_i) \mid \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 &= \langle \text{Definition of } \delta_I^+(c) \rangle \\
 & \prod \left\{ \prod \left\{ \bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq c \right\} \sqcup \bigsqcup_{i \in I} \mathfrak{s}_i(b_i) \mid \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 &= \langle \sqcup \text{ distributes over } \prod \rangle \\
 & \prod \left\{ \prod \left\{ \bigsqcup_{i \in I} \mathfrak{s}_i(a_i) \sqcup \bigsqcup_{i \in I} \mathfrak{s}_i(b_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq c \right\} \mid \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 &= \langle \text{Associativity of } \prod \rangle \\
 & \prod \left\{ \bigsqcup_{i \in I} (\mathfrak{s}_i(a_i) \sqcup \mathfrak{s}_i(b_i)) \mid \bigsqcup_{i \in I} a_i \sqsupseteq c \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 & \sqsupseteq \langle x \sqsupseteq y \text{ and } w \sqsupseteq z \text{ implies } x \sqcup w \sqsupseteq y \sqcup z; c_i = a_i \sqcup b_i; \mathfrak{s}_i(a_i) \sqcup \mathfrak{s}_i(b_i) = \mathfrak{s}_i(a_i \sqcup b_i) \rangle \\
 & \prod \left\{ \bigsqcup_{i \in I} \mathfrak{s}_i(c_i) \mid \bigsqcup_{i \in I} c_i \sqsupseteq c \sqcup d \right\} \\
 &= \langle \text{Definition of } \delta_I^+(c \sqcup d) \rangle \\
 & \delta_I^+(c \sqcup d)
 \end{aligned}$$

705 3. δ_I^+ is continuous.

706 Similar to proof of part (2). Since (Con, \sqsupseteq) is a finite scs, continuity follows from
707 preservation of finite joins.

708 (D.2) $\delta_I^+ = \mathfrak{s}_i$, if $I = \{i\}$.

709 Assume $I = \{i\}$ and let $c \in \text{Con}$.

$$\begin{aligned}
710 & \delta_{\{i\}}^+(c) \\
711 & = \langle \text{Definition of } \delta_{\{i\}}^+(c) \rangle \\
712 & \prod \left\{ \prod_{i \in \{i\}} \mathfrak{s}_i(a_i) \mid (a_i)_{i \in \{i\}} \in \text{Con}^{\{i\}} \text{ and } \prod_{i \in \{i\}} a_i \sqsupseteq c \right\} \\
713 & = \langle \text{Simplification} \rangle \\
714 & \prod \left\{ \mathfrak{s}_i(a_i) \mid a_i \in \text{Con} \text{ and } a_i \sqsupseteq c \right\} \\
715 & = \langle c \sqsupseteq c; \mathfrak{s}_i \text{ is monotonic} \rangle \\
716 & \mathfrak{s}_i(c)
\end{aligned}$$

718 (D.3) $\delta_I^+ \sqsupseteq_s \delta_J^+$, if $I \subseteq J$.

719 Assume $I \subseteq J$ and let $c \in \text{Con}$.

$$\begin{aligned}
720 & \delta_I^+(c) \\
721 & = \langle \text{Definition of } \delta_I^+(c) \rangle \\
722 & \prod \left\{ \prod_{i \in I} \mathfrak{s}_i(a_i) \mid (a_i)_{i \in I} \in \text{Con}^I \text{ and } \prod_{i \in I} a_i \sqsupseteq c \right\} \\
723 & \sqsupseteq \langle I \subseteq J; \prod_{i \in I} a_i \sqsupseteq \prod_{j \in J} a_j \rangle \\
724 & \prod \left\{ \prod_{j \in J} \mathfrak{s}_j(a_j) \mid (a_j)_{j \in J} \in \text{Con}^J \text{ and } \prod_{j \in J} a_j \sqsupseteq c \right\} \\
725 & = \langle \text{Definition of } \delta_J^+(c) \rangle \\
726 & \delta_J^+(c)
\end{aligned}$$

728 Therefore, $\delta_I^+(c) \sqsupseteq \delta_J^+(c)$ for all $c \in \text{Con}$.

729 Finally, from Theorem 21 since δ_I^+ is a gcd, then $\delta_I^+ \sqsubseteq_s \Delta_I$.

730 To complete the proof, we want to show that $\Delta_I(c) \sqsubseteq \delta_I^+(c)$ for all $c \in \text{Con}$. Let
731 $(a_i)_{i \in I} \in \text{Con}^I$ be an arbitrary tuple such that $\prod_{i \in I} a_i \sqsupseteq c$. Since $\{i\} \subseteq I$ for every i and Δ_I
732 is a gcd, $\Delta_I(c) \sqsubseteq \Delta_{\{i\}}(c) = \mathfrak{s}_i(c)$, thus $\prod_{i \in I} \Delta_I(c) \sqsubseteq \prod_{i \in I} \mathfrak{s}_i(c)$. Additionally, $\prod_{i \in I} \mathfrak{s}_i(c) \sqsubseteq$
733 $\prod_{i \in I} \mathfrak{s}_i(\prod_{i \in I} a_i)$ from monotonicity of every \mathfrak{s}_i . Therefore, $\Delta_I(c) \sqsubseteq \prod_{i \in I} \mathfrak{s}_i(\prod_{i \in I} a_i)$. Since
734 each \mathfrak{s}_i is continuous $\mathfrak{s}_i(\prod_{i \in I} a_i) = \prod_{i \in I} \mathfrak{s}_i(a_i)$, then $\Delta_I(c) \sqsubseteq \prod_{i \in I} \mathfrak{s}_i(a_i)$ for any $(a_i)_{i \in I} \in$
735 Con^I , thus $\Delta_I(c)$ is a lower bound of $\left\{ \prod_{i \in I} \mathfrak{s}_i(a_i) \mid (a_i)_{i \in I} \in \text{Con}^I \text{ and } \prod_{i \in I} a_i \sqsupseteq c \right\}$ and so
736 $\Delta_I(c) \sqsubseteq \delta_I^+(c)$ for every $c \in \text{Con}$. \blacktriangleleft

737 Proof of Theorem 30.

738 Let $(\Delta_I)_{I \subseteq G}$ be the distributed spaces of a finite scs $\mathbf{C} = (\text{Con}, \sqsubseteq, (\mathfrak{s}_i)_{i \in G})$. Suppose that
739 $(\text{Con}, \sqsubseteq)$ is a constraint frame. Let $J, K \subseteq G$ be two groups such that $I = J \cup K$. Then the
740 following equalities hold:

$$741 \quad 1. \Delta_I(c) = \prod \left\{ \Delta_J(a) \sqcup \Delta_K(b) \mid a \sqcup b \sqsupseteq c \right\}. \quad (4)$$

$$742 \quad 2. \Delta_I(c) = \prod \left\{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \right\}. \quad (5)$$

$$743 \quad 3. \Delta_I(c) = \prod \left\{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \mid a \sqsubseteq c \right\}. \quad (6)$$

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

$$745 \quad \Delta_J(a) = \prod \left\{ \prod_{j \in J} \mathfrak{s}_j(a_j) \mid (a_j)_{j \in J} \in \text{Con}^J \text{ and } \prod_{j \in J} a_j \sqsupseteq a \right\}$$

746 and

$$747 \quad \Delta_K(b) = \prod_{k \in K} \{ \bigsqcup_{k \in K} \mathfrak{s}_k(b_k) \mid (b_k)_{k \in K} \in \text{Con}^K \text{ and } \bigsqcup_{k \in K} b_k \sqsupseteq b \}$$

748 Thus, $\Delta_J(a) \sqcup \Delta_K(b) = \prod \{ \bigsqcup_{i \in I} \mathfrak{s}_i(c_i) \mid (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \}$ where for
749 every $\mathfrak{s}_j = \mathfrak{s}_k$, $\mathfrak{s}_i(c_i) = \mathfrak{s}_j(a_j) \sqcup \mathfrak{s}_k(b_k)$ and $c_i = a_j \sqcup b_k$. If $\mathfrak{s}_j \neq \mathfrak{s}_k$, either $\mathfrak{s}_i(c_i) = \mathfrak{s}_j(a_j)$
750 or $\mathfrak{s}_i(c_i) = \mathfrak{s}_k(b_k)$. Now, given $c \in \text{Con}$ consider the set $\{ \Delta_J(a) \sqcup \Delta_K(b) \mid a \sqcup b \sqsupseteq c \}$ for
751 any $a, b \in \text{Con}$.

$$\begin{aligned} 752 & \prod \{ \Delta_J(a) \sqcup \Delta_K(b) \mid a \sqcup b \sqsupseteq c \} \\ 753 & = \langle \text{Construction of } \Delta_J(a) \sqcup \Delta_K(b) \rangle \\ 754 & \prod \{ \prod_{i \in I} \{ \bigsqcup_{i \in I} \mathfrak{s}_i(c_i) \mid (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \} \mid a \sqcup b \sqsupseteq c \} \\ 755 & = \langle \text{Associativity of } \prod \rangle \\ 756 & \prod \{ \bigsqcup_{i \in I} \mathfrak{s}_i(c_i) \mid (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \text{ and } a \sqcup b \sqsupseteq c \} \\ 757 & = \langle (\sqsupseteq) x \sqsupseteq y \text{ and } w \sqsupseteq z \text{ implies } x \sqcup w \sqsupseteq y \sqcup z; (\sqsubseteq) \text{ construction of } c_i \rangle \\ 758 & \prod \{ \bigsqcup_{i \in I} \mathfrak{s}_i(c_i) \mid (c_i)_{i \in I} \in \text{Con}^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq c \} \\ 759 & = \langle \text{Lemma 28} \rangle \\ 760 & \Delta_I(c) \\ 761 & \end{aligned}$$

762 2. This property can be seen as a simplification of the first one: recall that $a \rightarrow c$ represents
763 the least element e such that $a \sqcup e \sqsupseteq c$. Take any b such that $a \sqcup b \sqsupseteq c$. Then $b \sqsupseteq a \rightarrow c$ and
764 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
765 follows that $\prod (S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c), \Delta_J(a) \sqcup \Delta_K(b) \}) = \prod (S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \})$
766 for any $S \subseteq \text{Con}$. This shows that $\Delta_J(a) \sqcup \Delta_K(b)$ is redundant since $\Delta_J(a) \sqcup \Delta_K(a \rightarrow c)$
767 is included in the set on the right-hand side of equality in the second property.

768 3. Similarly, this property can be seen as a simplification of the second one. Take any $a' \not\sqsubseteq c$.
769 It suffices to find $a \sqsubseteq c$ such that $\Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsupseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c)$ since then
770 $\prod (S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c), \Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \}) = \prod (S \cup \{ \Delta_J(a) \sqcup \Delta_K(a \rightarrow c) \})$ for
771 any $S \subseteq \text{Con}$.

772 Since $a' \not\sqsubseteq c$ either (a) $a' \sqsupset c$ or (b) a' and c are incomparable w.r.t. \sqsubseteq , written $a' \parallel c$.
773 Suppose (a) holds. Then take $a = c$ thus $a \rightarrow c = \text{true}$. By monotonicity we have
774 $\Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsupseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c)$ as wanted. Suppose that (b) $a' \parallel c$ holds.
775 Notice that $a' \rightarrow c \sqsubseteq c$. Suppose that $a' \rightarrow c = c$. Then we can take $a = \text{true}$, and thus
776 $a \rightarrow c = c = a' \rightarrow c$. By monotonicity we have $\Delta_J(a') \sqcup \Delta_K(a' \rightarrow c) \sqsupseteq \Delta_J(a) \sqcup \Delta_K(a \rightarrow c)$
777 as wanted. Suppose $a' \rightarrow c \sqsubset c$ holds. In this case, which is more interesting, we can
778 build a poset $L = (\{ a' \sqcup c, a', c, a' \rightarrow c, a' \sqcap (a' \rightarrow c) \}, \sqsubseteq)$ and verify that L is a
779 non-distributive sub-lattice of $(\text{Con}, \sqsubseteq)$, isomorphic to a lattice known as \mathbf{N}_5 (see Fig. ??).
780 But from order theory we know this cannot happen since we assumed $(\text{Con}, \sqsubseteq)$ to be
781 distributive, and distributive lattices do not have sub-lattices isomorphic to \mathbf{N}_5 ([3]).
782 ◀

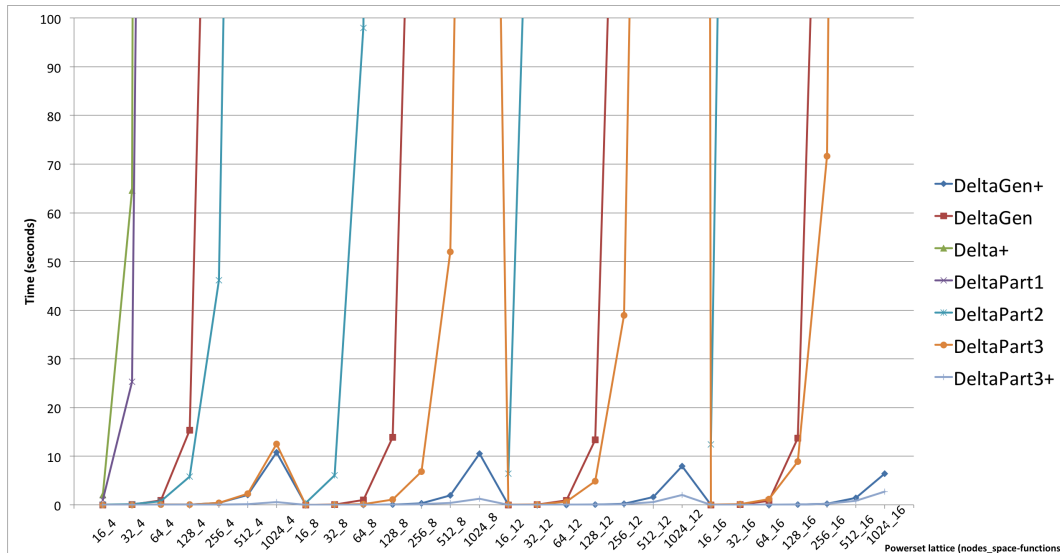
783 Proof of Complexity for Algorithms in Section 5.2.

784 The worst-case time complexity for computing the value $\Delta_I(c)$ for each algorithm is in (1)
785 $O(mn^m)$ for DELTA+, (2) $O(mn^{2 \log_2 m})$ for DELTAPART1, (3) $O(mn^{\log_2 m})$ for DELTAPART2,
786 DELTAPART3 and DELTAPART3+.

787 **Proof.** (1) DELTA+ checks n^m tuples with a cost $O(m)$ for a total of $O(mn^m)$. (2) The worst-
 788 case time complexity of DELTAPART1 is given by the recurrence equation $T(m) = n^2(1 +$
 789 $2T(m/2))$ and $T(1) = 1$ whose solution is $O(mn^{2 \log_2 m})$. (3) The worst-case time complexity
 790 of DELTAPART2, DELTAPART3 and DELTAPART3+ is given by $T(m) = n(1 + 2T(m/2))$ and
 791 $T(1) = 1$ whose solution is $O(mn^{\log_2 m})$.

792 The worst-case time complexity for computing function Δ_I with each of the algorithms
 793 above is obtained by multiplying the corresponding complexity by a factor of n . ◀

794 B Experimental Results



795 **Figure 4** Experimental results, average time, over n -element powerset lattices with a randomly
 796 generated number of space functions. Note that for the x -axis 512_8 means an 8-element powerset
 797 lattice with 512 nodes, and $|I| = 8$.

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

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

■ **Table 1** Average time in seconds for each algorithm over a powerset lattices with $|I| = 4$

805 **Index**

- 806 $(Con, \sqsubseteq, \mathfrak{s} = (\mathfrak{s}_i)_{i \in G})$, spatial constraint
807 system, scs, 5
- 808 G , group of agents, 5
- 809 \mathbf{C} , (Con, \sqsubseteq) , constraint system, cs, 4
- 810 Con , set of constraints, 4
- 811 Δ , distributed spaces, 8
- 812 Δ_I , distributed space of a group I , 8
- 813 $\Pi_I(c)$, the I -group projection of $c \in Con$, 10
- 814 \sqsubseteq , entailment, order relation, 3
- 815 $\sim \cdot$, Heyting negation, 4
- 816 δ , group distribution candidate, 9
- 817 *false* (top), all (possibly inconsistent)
818 information, 3
- 819 $\lambda_{\perp}(c)$, biggest space, bottom element of the
820 lattice, 8
- 821 $\lambda_{\top}(c)$, smallest space, empty space, top
822 element of the lattice, 8
- 823 $\mathcal{S}(\mathbf{C})$, set of all space functions over a cs \mathbf{C} , 7
- 824 $\pi_I(c)$, I -join projection of a group I of c , 6
- 825 $\pi_i(c)$, i -agent projection of $c \in Con$, 6
- 826 \rightarrow , Heyting implication, 4
- 827 $\mathcal{S}(\mathbf{C})$, set of all space functions over
828 $\mathbf{C} = (Con, \sqsubseteq)$, 5
- 829 \mathfrak{s} , space tuple, 5
- 830 \mathfrak{s}_i , space function of agent i , 5
- 831 $\sqcap, \sqbar{\sqcap}$, meet, GLB, infimum, 3
- 832 $\sqcup, \sqbar{\sqcup}$, join, LUB, supremum, 3
- 833 *max* max operation, 3
- 834 *true* (bottom), empty information, 3
- 835 X^J , denotes the set of tuples $(x_j)_{j \in J}$ of
836 elements $x_j \in X$ for each $j \in J$,
837 12
- 838 $f \sqsubseteq_s g$, space order, 7
- 839 compact element, 3
- 840 constraint frame, 4
- 841 continuous function, 3
- 842 D.1, first axiom group distribution candidate,
843 9
- 844 D.2, second axiom group distribution
845 candidate, 9
- 846 D.3, third axiom group distribution
847 candidate, 9
- 848 directed set, 3
- 849 finite scs, 11
- 850 join-homomorphism, 3
- 851 join-preservation, 3
- 852 lattice, complete, 3
- 853 S.1, first space axiom, 5
- 854 S.2, second space axiom, 5