On Extending Brandt’s Speedup Theorem from LOCAL to Round-Based Full-Information Models

Paul Bastide  
Ecole Normale Supérieure de Rennes, France.  
paul.bastide@ens-rennes.fr

Pierre Fraigniaud  
Université de Paris and CNRS, France. Additional supports from ANR Projects DESCARTES and DUCAT.  
pierre.fraigniaud@irif.fr

Abstract  
Given any task \(\Pi\), Brandt’s speedup theorem (PODC 2019) provides a mechanical way to design another task \(\Pi'\) on the same input-set as \(\Pi\) such that, for any \(t \geq 1\), \(\Pi\) is solvable in \(t\) rounds if and only if \(\Pi'\) is solvable in \(t - 1\) rounds. The theorem applies to the anonymous variant of the LOCAL model, in graphs with sufficiently large girth, and to locally checkable labeling (LCL) tasks. In this paper, using combinatorial topology applied to distributed computing, we dissect the construction in Brandt’s speedup theorem for expressing it in the broader framework of round-based models supporting full information protocols, which includes models as different as wait-free shared-memory computing with iterated immediate snapshots, and synchronous failure-free network computing. In particular, we provide general definitions for notions such as local checkability and local independence, in our broader framework. In this way, we are able to identify the hypotheses on the computing model, and on the tasks, that are sufficient for Brandt’s speedup theorem to apply. More precisely, we identify which hypotheses are sufficient for the each direction of the if-and-only-if condition. Interestingly, these hypotheses are of different natures. Our general approach enables to extend Brandt’s speedup theorem from LOCAL to directed networks, to hypergraphs, to dynamic networks, and even to graphs including short cyclic dependencies between processes (i.e., the large girth condition is, to some extent, not necessary). The theorem can even be extended to shared-memory wait-free computing. In particular, we provide new impossibility proofs for consensus and perfect renaming in 2-process systems.

2012 ACM Subject Classification  
Theory of computation \(\rightarrow\) Distributed algorithms

Keywords and phrases  
Local Checkability; Distributed Complexity and Computability.

Digital Object Identifier  
10.4230/LIPIcs...
In particular, $\Pi^{(t)}$ is solvable in zero rounds, and demonstrating that $\Pi^{(t)}$ is actually not solvable in zero rounds establishes the lower bound $t + 1$ for the round-complexity of $\Pi$.

Brandt’s speedup theorem does not directly applies to LOCAL, but to an anonymous variant of LOCAL on graphs with sufficiently large girth. This is because the presence of identifiers assigned to the nodes prevents local-independence to be satisfied, where the latter is a property that is essential for establishing the theorem. It is not trivial to formally express this property, but, roughly speaking, given the radius-$(t - 1)$ views of two adjacent nodes $v$ and $v'$ in some network $G$, the presence of identifiers results in the fact that one cannot guarantee that two independent extensions of these two views into radius-$t$ views are compatible. Indeed, one extension may include a node $w$ provided with the same identifier as a node $w'$ in the other extension, with $w \neq w'$, in contradiction with the fact that each identifier must be unique in the network. Local independence also imposes to consider graphs $G$ with girth $g > 2t - 1$. Indeed, in graphs with girth $g \leq 2t - 1$, two independent radius-$t$ extensions of the radius-$(t - 1)$ views of $v$ and $v'$ may include a same node $w$ provided with different identifiers, or with different inputs. This would result into two non-compatible radius-$t$ extensions in the sense that there are no instances yielding the simultaneous presence of these two radius-$t$ views at two adjacent nodes.

Also, Brandt’s speedup theorem requires the tasks at hand to be locally checkable. This property essentially says that, given an assignment of input-output values to the nodes, the correctness of the collection of output values with respect to the collection of input values can be established by merely inspecting the values of each node and of its neighbors in the network. In other words, a task is locally checkable if the correctness of an assignment of values to the nodes is defined as the conjunction of the local correctness of this assignment, where “local” refers to the closed neighborhood of each node. Proper coloring and maximal independent set (MIS) are typical examples of locally checkable tasks in LOCAL.

We can now rephrase our original questioning in the specific case of Brandt’s speedup theorem: does this theorem holds in other models? For such a question to make sense, we restrict attention to models in which the notion of rounds is defined, which naturally include synchronous models in networks with multiparty interactions, namely hypergraphs, and synchronous models in networks that evolve with time, namely, dynamic networks. Round-based models however include far more than just synchronous models in networks. For instance, asynchronous shared-memory computing with iterated immediate snapshots, referred to as WAIT-FREE in the following, which is computationally equivalent to asynchronous read/write shared-memory computing with crash-prone processes, is round-based. The same holds for $t$-resilient computing, $0 \leq t \leq n - 1$, which is essentially the same as WAIT-FREE, but where at most $t$ processes can crash.

The LOCAL model has another feature. It supports full information communication protocols. That is, whenever a process receives information from another process, one can assume that the latter has sent all the data it acquired before the communication took place. This assumption enables the design of strong lower bounds, which hold even if the processes are not restricted in term of volume of communication. Also, the LOCAL model does not restrict the individual computational power of the processes. This assumption enables the design of unconditional lower bounds, which hold independently from complexity or computability assumptions regarding the computing power of each individual process. All the models mentioned above support full-information protocols, and have unlimited

---

1 The notion of local checkability can be extended to neighborhood at distance $k$ in a straightforward manner, for any fixed $k \geq 1$. 
individual computational power.

So, making our questioning even more specific: Is there an analog of Brandt’s speedup theorem for all round-based models supporting full-information protocols with unlimited individual computational power? If not, what make the LOCAL model so special? If yes, for which models? Under which conditions?

**Our Results.** Using the framework provided by combinatorial topology applied to distributed computing, we give a general definition of speedup tasks for round-based models supporting full-information protocols (with unlimited individual computational power). Given a task $\Pi$ in the LOCAL model, Brandt’s speedup theorem constructs such a speedup task $\Pi' = \Phi(\Pi)$. We then revisit Brandt’s construction, that is, we dissect the nature of the operator $\Phi$ transforming any task $\Pi$ into a task $\Pi' = \Phi(\Pi)$, for identifying the central assumptions allowing this construction to work in LOCAL. They are two central assumptions: local checkability and local independence. We extend these two notions from the LOCAL model to round-based models supporting full-information protocols. We also extend Brandt’s operator $\Phi$ to all such models. We denote by $\Phi^*$ this extension. As a result, we are able to express a general speedup theorem, which roughly reads as follows. Let $\mathcal{M}$ be a round-based model supporting full-information protocols, let $\Pi$ be a task, and let $t \geq 1$. The task $\Phi^*(\Pi)$ satisfies the following:

1. Assume that $\Pi$ satisfies $(t-1)$-independence with respect to $\mathcal{M}$. If $\Pi$ is solvable in at most $t$ rounds, then $\Phi^*(\Pi)$ is solvable in at most $t-1$ rounds.
2. Assume that $\Pi$ is locally checkable in $\mathcal{M}$. If $\Phi^*(\Pi)$ is solvable in at most $t-1$ rounds, then $\Pi$ is solvable in at most $t$ rounds.

Statement 1 guarantees that the task $\Phi^*(\Pi)$ is at least “1-round faster” than the original task $\Pi$. Statement 2 guarantees that $\Phi^*(\Pi)$ is no more than “1-round faster”, and in particular that $\Phi^*(\Pi)$ is not solvable in zero rounds. Observe that the sets of hypotheses required for each of the two statements are different, and actually they do not even intersect. This provides flexibility. For instance, given a task $\Pi$ satisfying local independence w.r.t. model $\mathcal{M}$, even if $\Pi$ is not locally checkable in $\mathcal{M}$, it may still be the case that, thanks to Statement 1, $\Phi^*(\Pi)$ remains solvable in at least $f(t)$ rounds for some function $f$, and that iterating $\Phi^*$ results in a non-trivial lower bound. For instance, $f(t) = \log t$ is sufficient for deriving a lower bound $\Omega(\log^* n)$. Satisfying Statement 2 requires to limit the class of tasks under consideration to locally checkable tasks. For instance, proper-coloring and renaming are locally checkable in LOCAL and WAIT-FREE, respectively, but spanning tree and consensus are not locally checkable in these respective models.

Concretely, our general construction $\Phi^*$ allows us to directly extend Brandt’s speedup theorem to various kinds of synchronous models in networks, including directed graphs, hypergraphs, dynamic networks, and even to graphs including short cyclic dependencies between processes (i.e., the large girth condition is, to some extent, not necessary). Interestingly, our general construction also enables to extend Brandt’s speedup theorem to asynchronous failure-prone computing models such as WAIT-FREE. In particular, we provide a new impossibility proof for consensus and for perfect renaming in 2-process systems.

**Related Work.** Roughly, the modern approach of distributed computing can be presented as the study of two large classes of computing models, one whose models are aiming at capturing issues related to *time* (asynchrony, crashes, etc.) [3], and another whose models are aiming at capturing issues related to *space* (latency, congestion, etc.) [36]. The study of the
former class puts emphasis on the study of system tasks such as leader election or consensus, while study of the latter class put emphasis on the study of graph problems such as coloring or matching. The applications of topology to the theory of distributed computing was introduced in \cite{28,38} for studying system tasks under asynchronous crash-prone computing models. This paper is inspired from \cite{28}, which identified the properties of the topological deformations related to wait-free and $t$-resilient computing. Since then, topology has been extensively used in the context of asynchronous computing with crash-prone processes, for establishing lower bounds or impossibility results \cite{2,10,22}, but also upper bounds \cite{11}. It has also been extended to mobile computing \cite{1}, to dynamic environments \cite{23}, and to Byzantine failures \cite{34}. Moreover, a topological description of concurrent programming has been developed \cite{15,25}. It is however only recently that distributed network computing has been approached through the lens of combinatorial topology \cite{9,20}, specifically applied to local computing.

The \textsc{LOCAL} model is a synchronous failure-free model dedicated to capture local computing in networks, that is, the ability to solve problems by having each process inspecting solely the inputs present in its vicinity in the network (see \cite{29,36}). Among the earliest seminal work in the \textsc{LOCAL} model are \cite{33} and \cite{35}. The former established the celebrated lower bound $\Omega(\log^* n)$ rounds for 3-coloring the $n$-node cycle. The latter introduced the class of \textit{locally checkable labeling} (LCL) problems, that is, the class of problems defined on bounded-degree graphs, involving individual inputs and outputs of bounded size, and whose candidate solutions can be checked locally\footnote{The class of tasks that are locally checkable are sometimes referred to as “the equivalent of NP”. This is however debatable, as the formal definitions of complexity classes such as NP typically involve \textit{proofs} provided by non-trustable oracles. In the context of distributed network computing, the “equivalents of NP” may rather be the classes PLS, LCP, and NLD, respectively defined in \cite{26,24,19}.}. It was shown that it is undecidable whether a given LCL problem is solvable locally (i.e., in a constant number of rounds). It was also shown that if an LCL problem can be solved locally by a randomized algorithm, then it can be solved locally by a deterministic algorithm. This derandomization result initiated a vast literature on the power and limitation of randomized algorithms in their ability of solving problems locally (see, e.g., \cite{7,14,37} for recent contributions). The aforementioned reference \cite{33} introduced a lower bound technique bearing similarities with the topological approach, that connects a structural property of a graph capturing all possible configurations of the system at a given time $t$ with the ability to solve a problem in $t$ rounds. For a quarter of a century, this was the only known non-trivial lower bound technique departing from using indistinguishably arguments, until the breakthrough \cite{8} introducing the aforementioned speedup technique. This technique, designed for LCL problems in general, was successfully applied for deriving lower bounds on various problems such as sinkless orientation and 2-weak coloring \cite{8}, as well as maximal matching and maximal independent set \cite{6}. We refer to the recent paper \cite{39} for more details on using the speedup technique to understand locality.

The study of distributed algorithms for networks has recently been subject to generalizations from graphs to hypergraphs, for handling frameworks with multiparty interactions. In particular, the maximal independent set problem in hypergraphs was studied in \cite{32}, and maximal matching in hypergraphs was studied in \cite{17}. Interestingly, that latter paper shows that solving problems on hypergraphs has also surprising implications on solving other problems efficiently on graphs. Various extensions of the maximal independent set problem were studied in hypergraphs in \cite{31}, specifically for linear hypergraphs (i.e., hypergraphs in which any two hyperedges overlap on at most one node). We refer to this latter paper for
Figure 1 The topological approach of distributed computing

pointers on earlier contributions on the design of distributed algorithms for hypergraphs.

2 Summary of our Contributions

This section is a technical summary of our approach and main results. The model considered in this section is not the most general one, and our general model will be introduced further in the paper. In particular, the model presented in this section does not capture hypergraphs. Nevertheless, it is sufficient for presenting our main ideas and techniques.

2.1 Distributed Computation

Combinatorial topology provides an elegant and unified way to describe distributed computing (see [27,28]). We refer to Section 3 for the details, but, roughly, a task (e.g., consensus, vertex-coloring, renaming, maximal independent set, etc.) for an n-process system with processes $p_1, \ldots, p_n$ can be defined as a triple $(I, O, \Delta)$, where $I$ and $O$ respectively denote the sets of all legal $k$-process input and output states, $1 \leq k \leq n$, and $\Delta : I \rightarrow 2^O$ is a function that maps every input state $\sigma \in I$ to the set of output states $\Delta(\sigma)$ that are legal w.r.t. $\sigma$. See Fig. 1.

Formally, $I$ and $O$ are simplicial complexes, and every set $\sigma$ in one of these two complexes is a simplex (see Appendix A.1 for a brief introduction to combinatorial topology). A simplex $\sigma \in I$ (resp., $\sigma \in O$) is of the form $\sigma = \{(i, x_i) : i \in I\}$, where (1) $I \subseteq [n]$ is any non-empty set of process names, (2) for every $i \in I$, $x_i$ is an input (resp., output) value, and (3) these values are mutually compatible whenever, for every $i \in I$, $x_i$ is assigned to $p_i$. A 0-dimensional simplex $\{(i, x_i)\}$, for some $i \in [n]$, is called a vertex. It is assumed that $\Delta$ preserves names, that is, if $\tau \in \Delta(\sigma)$, then name($\tau$) = name($\sigma$) = $I$. We denote by $\text{val}(I)$ and $\text{val}(O)$ the set of input and output values, respectively. That is, for $\sigma = \{(i, x_i) : i \in I\}$, if $\sigma \in I$ (resp., $\sigma \in O$) then $x_i \in \text{val}(I)$ (resp., $x_i \in \text{val}(O)$) for every $i \in I$.

We consider any round-based communication model $M$ supporting full information protocols, which include, e.g., wait-free computing with iterated immediate snapshots, referred to as WAIT-FREE [3][27], in the context of shared-memory computing, and LOCAL in the context of network computing [36]. A crucial feature shared by all these models is that, w.l.o.g., one can restrict attention to algorithms decomposed into two phases: one phase consisting of a certain number $t$ of communication rounds where, at each round, each process forwards all the information acquired during the previous rounds, and one phase of computation in which an output is computed based on all the information accumulated during the $t$ communication rounds performed during the first phase. So, in fact, designing a $t$-round algorithm boils down to designing an output function mapping views gathered within $t$ communication rounds to output values.

Let $P^{(t)}$ denote the set of all possible $k$-process states of the system, $1 \leq k \leq n$, after $t$ rounds, with $P^{(0)} = I$. Like $I$ and $O$, $P^{(t)}$ is a simplicial complex, for every $t \geq 0$. For $t > 0$, the complex $P^{(t)}$ is the image of $P^{(t-1)}$ by a function $\Xi$, which is specific of
The communication model $\mathcal{M}$, and which is mapping every state $\sigma \in \mathcal{P}^{(t-1)}$ to the set $\Xi(\sigma) \subseteq \mathcal{P}^{(t-1)}$ of states that may result from $\sigma$ after one round of communication. As for the input-output specification $\Delta$, if $\tau \in \Xi(\sigma)$ then name($\tau$) = name($\sigma$). Note that, in particular, $\mathcal{P}^{(1)} = \Xi'(I)$, as displayed on Fig. 1. For instance, given $\sigma = \{(i, x_i) : i \in I\} \in \mathcal{P}^{(t-1)}$, we have:

- In WAIT-FREE, $x_i$ is the view of $p_i$ resulting from its $(t-1)$th snapshots. We have $\tau \in \Xi(\sigma)$ if $\tau = \{(i, \{x_j : j \in J_i\}) : i \in I\}$ where, for every $i \in I$, (1) $J_i \subseteq I$, (2) $J_i \subseteq J_j$ or $J_i \supseteq J_j$ for every $j \in I$, and (3) for every $j \in I$, if $j \in J_i$ then $J_j \subseteq J_i$.

- In LOCAL, $x_i$ is the labeled ball of radius $t-1$ centered at $p_i$ in the input graph $G$.

Assuming $I = [n], \tau \in \Xi(\sigma)$ if $\tau = \{(i, \{x_j : j \in N_G[i]\}) : i \in [n]\}$ where $N_G[i]$ denotes the closed neighborhood of node $i$ in the underlying network $G$.

More generally, we model a communication model $\mathcal{M}$ as a simplicial complex whose simplices are of the form $\varphi = \{(i, J_i) : i \in I\}$ with $i \in J_i \subseteq [n]$ for every $i \in I$. Such a simplex corresponds to a possible communication round in which, for every $i \in I$, process $i$ receives information from all processes $j \in J_i$. See Figs. 2 and 3 for examples in LOCAL and WAIT-FREE. A simplex $\varphi = \{(i, J_i) : i \in I\}$ of $\mathcal{M}$ is said to be closed if $\cup_{i \in I} J_i = I$. To every communication model $\mathcal{M}$ corresponds a communication map $\Xi$. Let $\sigma = \{(i, v_i) : i \in I\} \in \mathcal{P}^{(t)}$ for some $t \geq 0$, where $I \subseteq [n]$, and let us assume that there exists a closed simplex $\varphi = \{(i, J_i) : i \in I\}$ in $\mathcal{M}$. We set $\Xi(\sigma, \varphi) = \{(i, \{v_j : j \in J_i\}) : i \in I\}$, and we define $\Xi(\sigma) = \{\Xi(\sigma, \varphi) : (\varphi \in \mathcal{M}) \land (\text{name}($\varphi$) = \text{name}($\sigma$)) \land (\varphi \text{ is closed})\}$.

A $t$-round algorithm is then a function $\delta : \mathcal{P}^{(t)} \rightarrow \mathcal{O}$ mapping every pair $(i, x_i) \in \mathcal{P}^{(t)}$ to some pair $(i, y_i) = \delta(i, x_i) \in \mathcal{O}$ (cf. Fig. 1). Note that $\delta$ is name-preserving. The semantic of this map is that process $i$ in state $x_i$ outputs $y_i$. In wait-free computing, $\delta$ essentially takes views resulting from $t$ rounds of iterated immediate snapshots as inputs, while, in LOCAL, $\delta$ takes labeled balls of radius $t$ as inputs. The function $\delta$ must satisfy two constraints:

- $\delta$ is simplicial, that is, for every $\sigma = \{(i, x_i) : i \in I\} \in \mathcal{P}^{(t)}$, $\delta(\sigma) = \{(i, x_i) : i \in I\} \in \mathcal{O}$, i.e., $\delta(\sigma)$ is a legal $k$-process output state, where $k = |I|$, and

- $\delta$ agrees with $\Delta$, that is, given any input state $\sigma \in \mathcal{I}$, $\delta(\Xi(\sigma)) \subseteq \Delta(\sigma)$, i.e., the output after $t$ rounds of a set of processes initially in state $\sigma \in \mathcal{I}$ must be one of the output states that are legal w.r.t. $\sigma$. 

Figure 2: The communication model is the (synchronous failure-free) directed cycle $C_3$. In the considered task, $p_1$ receives input $B$ (blue) as input, while $p_2$ and $p_3$ may receive input $R$ (red) or $G$ (green). The output complex specifies that the output values must be a proper 3-coloring of $C_3$, with $p_1$ colored blue. After one round, the state of each process is a pair $XY$ of colors, where $X$ is its input color, and $Y$ is a color received from the in-neighbor in $C_3$, forming the protocol complex $\mathcal{P}^{(1)}$. As an input-output specification $\Delta$, we consider the case where $p_2$ must output the same color as its input color. The map $\delta : \mathcal{P}^{(1)} \rightarrow \mathcal{O}$ depicted on the figure is simplicial and agrees with $\Delta$. Note that $\delta$ is also name-independent. There are no simplicial maps from $\mathcal{I} = \mathcal{P}^{(0)}$ to $\mathcal{O}$ that agree with $\Delta$, and therefore the task is not solvable in zero rounds (even if one discards name-independence). 

Network Computing

Communication model $\mathcal{M}$

Input complex $\mathcal{I}$

Protocol complex $\mathcal{P}^{(1)}$

Output complex $\mathcal{O}$
The same holds for any theorem.

only if there exists a simplicial map \( \beta : \Xi(\Delta) \to O \) such that, for every \( \sigma \in I, \beta(\Delta'(\sigma)) \subseteq \Delta(\sigma) \).

Note that \( \Xi(O') \) is the set of all global states that may result after a single round of communication under \( M \), starting from input states in \( O' \). The terminology “speedup task” is motivated by the following simple observation.

Lemma 1. A task \((I, O, \Delta)\) is solvable in at most \( t \) rounds if and only if there exists a simplicial map \( \delta : \mathcal{P}^{(t)} \to O \) that agrees with \( \Delta \).

Depending on the context, one usually requires that \( \delta \) is name-independent, that is, if \( \delta(i, x) = (i, y) \) and \( \delta(j, x) = (j, y') \), then \( y = y' \), reflecting the fact that the name of a process is external, and not part of its input. See Figs. 2 and 3 for examples.

2.2 Speedup Tasks

In this section, we introduce a general notion of speedup tasks. See Section 4 for more details.

Definition. The speedup of a task \((I, O, \Delta)\) for a (full information) communication model \( M \) is a task \((I, O', \Delta')\) such that (see Fig. 4):

- for every \( t \geq 1 \), if there exists a simplicial map \( \delta : \mathcal{P}^{(t)} \to O \) that agrees with \( \Delta \), then there exists a simplicial map \( \alpha : \mathcal{P}^{(t-1)} \to O' \) that agrees with \( \Delta' \), and
- there exists a simplicial map \( \beta : \Xi(O') \to O \) such that, for every \( \sigma \in I, \beta(\Delta'(\sigma)) \subseteq \Delta(\sigma) \).

Note that \( \Xi(O') \) is the set of all global states that may result after a single round of communication under \( M \), starting from input states in \( O' \). The terminology “speedup task” is motivated by the following simple observation.

Lemma 2. For every \( t \geq 1 \), there is a \( t \)-round algorithm for \((I, O, \Delta)\) in model \( M \) if and only if there exists a \((t-1)\)-round algorithm for its speed up task \((I, O', \Delta')\) in model \( M \).

Indeed, the simplicial map \( \alpha \) guarantees the solvability of \((I, O, \Delta)\) in \( t-1 \) rounds assuming that \((I, O, \Delta)\) is solvable in \( t \) rounds, and the simplicial map \( \beta \) guarantees the solvability of \((I, O', \Delta')\) in \( t \) rounds assuming that \((I, O', \Delta')\) is solvable in \( t-1 \) rounds.

Generic Approach for Constructing Speedup Tasks. There is a generic approach for constructing speedup tasks. To see how, note that, given its state \( v_i \) after \( t-1 \) rounds, every process \( p_i \) can internally build all its possible futures after one more round, as well as all the possible futures of all the other processes, whenever the current states of the other processes after \( t-1 \) rounds are compatible with \( p_i \) in state \( v_i \). Let \( \text{St}(i, v_i) \) denote the star of \((i, v_i)\) in \( \mathcal{P}^{(t-1)} \), that is, the set of simplices of \( \mathcal{P}^{(t-1)} \) containing \((i, v_i)\) (see Fig. 5(a)). Let
On Extending Brandt’s Speedup Theorem

XX:8

Cl(St(i, v_i)) be the closure of this star, that is, the minimal complex containing all simplices in St(i, v_i).

The complex Cl(St(i, v_i)) precisely captures all the possible states of the system after round t − 1, given that process i is in state v_i. Process i in state v_i can then compute Ξ(Cl(St(i, v_i))), by simulating all possible scenarios resulting from one more round of communication. It follows that, after round t − 1, process i in state v_i can output

\[ α(i, v_i) = δ \( \Xi (\text{Cl}(\text{St}(i, v_i))) \) \]

Observe that, for every \((i, v_i) \in \mathcal{P}^{t-1}\), \(α(i, v_i)\) is a subcomplex of the output complex \(\mathcal{O}\) of the task \((\mathcal{I}, \mathcal{O}, \Delta)\) at hand. This provides us with the intuition that, if the task \((\mathcal{I}, \mathcal{O}, \Delta)\) has a speedup task \((\mathcal{I}, \mathcal{O}', \Delta')\), the complex \(\mathcal{O}'\), as well as the input-output specification \(\Delta'\) are in close connection with the subcomplexes of \(\mathcal{O}\), with the idea in mind that every process can extrapolate its current state by simulating all possible scenarios resulting from one more round of communication. In short, we foresee that a vertex of \(\mathcal{O}'\) is of the form \((i, K)\) where \(i \in [n]\), and \(K \subseteq \mathcal{O}\) is a complex. Since a simplicial complex is a collection of sets of values, this provides us with the intuition that an output value for the speedup task \((\mathcal{I}, \mathcal{O}', \Delta')\) is a set of set of output values in \(\mathcal{O}\). Now, the next question to address is, what are the consistency conditions to be satisfied for a set \((\{i, K_i\} : i \in I)\) of vertices of \(\mathcal{O}'\) to be a simplex of \(\mathcal{O}'\)? At this stage of the discussion, it is not yet clear what these conditions should look like, but one can identify one hypothesis that helps very much, called local independence. Roughly, given a simplex \((\{i, v_i\}, \{j, v_j\}) \in \mathcal{P}^{t-1}\), we would like all simplices in \(α(i, v_i) = δ(\Xi(\text{Cl}(\text{St}(i, v_i))))\) and \(α(j, v_j) = δ(\Xi(\text{Cl}(\text{St}(j, v_j))))\) to be compatible.

The Local Independence Property. Let \(t \geq 1\) be an integer. A task \((\mathcal{I}, \mathcal{O}, \Delta)\) satisfies the \(t\)-independence property w.r.t. a communication model \(\mathcal{M}\) if, for every closed simplex \(\varphi = \{(i, J_i) : i \in I]\) \(\in \mathcal{M}\), for every \(i \in I\), for every \(j \in J_i\), for every simplex \((\{i, v_i\}, \{j, v_j\}) \in \mathcal{P}^{t}\), and for every two collections of dimension-1 simplices

\[
\begin{align*}
C_i &= \{\sigma_k = \{(i, v_i), (k, v_k)\} \in \text{St}(i, v_i) : k \in J_i \setminus \{i, j\}\} \\
C_j &= \{\sigma_k = \{(j, v_j), (k, v_k)\} \in \text{St}(j, v_j) : k \in J_j \setminus \{i, j\}\}
\end{align*}
\]

we have \(\bigcup_{\sigma \in C_i \cup C_j} \sigma \in \mathcal{P}^{t}\).

Note that the \(t\)-independence property depends solely of the model \(\mathcal{M}\), and of the input complex \(\mathcal{I}\) of the task. Indeed, \(\mathcal{I}\) and \(\mathcal{M}\) are the only parameters that govern the properties of the protocol complex at time \(t\). An interesting special case of the \(t\)-independence property is when considering \(j = i\). The \(t\)-independence property then implies that, for every \((i, J_i) \in \mathcal{M}\), for every vertex \((i, v_i) \in \mathcal{P}^{t}\), and for every collection \(C = \{\sigma_k = \{(i, v_i), (k, v_k)\} \in \text{St}(i, v_i) : k \in J_i \setminus \{i\}\}\), we have \(\bigcup_{\sigma \in C} \sigma \in \mathcal{P}^{t}\).

Note that any task \((\mathcal{I}, \mathcal{O}, \Delta)\) for two processes satisfies the \(t\)-independence property w.r.t. any communication model \(\mathcal{M}\), for any \(t \geq 0\). The task depicted on Fig. 4 satisfies

\[
\begin{array}{c}
\mathcal{P}^{t-1} \xrightarrow{\Xi^{t-1}} \mathcal{I} \xrightarrow{\Xi^1} \mathcal{P}^{t} \xrightarrow{α \Delta'} \mathcal{O}' \xrightarrow{β \Delta} \mathcal{O} \xrightarrow{\delta}
\end{array}
\]

**Figure 4** The task \((\mathcal{I}, \mathcal{O}', \Delta')\) is a speedup task for \((\mathcal{I}, \mathcal{O}, \Delta)\)
The two edges skeleton star is the complex obtained by adding the four vertices \( (2, RB), (3, RR), (3, RG), \) and \( (2, GB) \), plus the two edges \( (2, RB), (3, RR) \) and \( (2, GB), (3, RG) \). (b) The skeleton \( \text{Sk}(1,2)(\mathcal{P}^{(1)}) \). (c) The skeleton \( \text{Sk}(2)(\mathcal{O}) \).

0-independence w.r.t. \( \widetilde{C}_3 \) because, for every \( \{(i, v_i), (j, v_j)\} \in \mathcal{P}^{(0)} = \mathcal{I} \), either \( J_i = \{i, j\} \) or \( J_j = \{i, j\} \), and any input value \( v_k \) of the third process \( p_k \) is compatible with \( v_i \) and \( v_j \), i.e., \( \{(i, v_i), (j, v_j), (k, v_k)\} \in \mathcal{P}^{(0)} \). On the other hand, it is not 1-independent. To see why, let us consider the simplex \( \{(1, BR), (2, RB)\} \in \mathcal{P}^{(1)} \). We have \( \{(1, BR), (3, RG)\} \in \text{St}(1, BR) \), but \( \{(1, BR), (2, RB), (3, RG)\} \notin \mathcal{P}^{(1)} \). Yet, every task with locally checkable inputs satisfies the \( t \)-independence property w.r.t. the anonymous variant of \( \text{LOCAL} \) in graph with girth larger than \( 2t + 1 \). Indeed, in \( \text{LOCAL} \) the values \( v_i \) in \( \mathcal{P}^{(t)} \) are input-labeled balls of radius \( t \) centered at \( p_i \), and \( t \)-independence boils down to the ability to extend these balls into balls of radius \( t + 1 \) in a compatible manner for any two adjacent processes. See [8] for more details.

2.3 A General Construction of Speedup Tasks

Let \( (\mathcal{I}, \mathcal{O}, \Delta) \) be a task for \( n \) processes. For defining a speedup task \( (\mathcal{I}, \mathcal{O}', \Delta') \), we define the complex \( \mathcal{O}' \), and the input-output specification \( \Delta' \), as follows. The construction is inspired by the aforementioned map \( \alpha : \mathcal{P}^{(t-1)} \rightarrow \mathcal{O}' \) defined by \( \alpha(i, v_i) = \delta \left( \Xi \left( \text{Cl}(\text{St}(i, v_i)) \right) \right) \), but every complex resulting from the application of \( \alpha \) at a vertex \( (i, v_i) \) of \( \mathcal{P}^{(t-1)} \) is decomposed into sets of vertices. More specifically, let \( I \subseteq [n] \). We denote by \( \text{Sk}_I(\mathcal{O}) \) the skeleton of \( \mathcal{O} \) composed of all simplices \( \sigma \in \mathcal{O} \) with \( \text{name}(\sigma) \subseteq I \) (see Fig. 5(b)). In particular, \( \text{Sk}_{\{i\}}(\mathcal{O}) \) is merely a set of vertices of \( \mathcal{O} \), each of the form \( (i, y) \) for some \( y \in \text{val}(\mathcal{O}) \) (see Fig. 5(c)). First, we describe the vertices of \( \mathcal{O}' \), then its simplices, and finally the input-output specification \( \Delta' \). See Section 5 for more details.

Vertices of \( \mathcal{O}' \). Each vertex in \( \mathcal{O}' \) is a pair \( (i, \mathbb{P}_i) \) with \( \mathbb{P}_i = (x_i, S_i) \), \( x_i \in \text{val}(\mathcal{I}) \), and

\[
S_i = \{ S_{i,k,J_i} : (i, J_i) \in \mathcal{M} \text{ and } k \in [n] \},
\]

where, for every vertex \( (i, J_i) \in \mathcal{M} \), and for every \( k \in [n] \), \( S_{i,k,J_i} = 2^{\mathcal{A}(i)^{\mathcal{O}'}} \). In other words, each \( S_{i,k,J_i} \) is a collection of sets with elements in \( \text{Sk}_{\{i\}}(\mathcal{O}) \). There is a set \( S_{i,k,J_i} \) for every set \( J_i \) of processes from which process \( i \) may receive information in some communication specified by \( \mathcal{M} \), and for every process \( k \in [n] \). Each set \( S_{i,k,J_i} \in S_i \) is identified in \( S_i \) by a pair \( (k, J_i) \). That is, formally, \( S_i \) is an array indexed by pairs (process, set of processes).

Nevertheless, for the sake of simplifying the notation, we describe \( S_i \) as a set. For a pair \( (i, (x_i, S_i)) \) to be a vertex of \( \mathcal{O}' \), the sets \( S_{i,k,J_i} \) in \( S_i \) must satisfy the following property:

\( P0 \): For every vertex \( (i, J_i) \in \mathcal{M} \) with \( J_i = \{ k_1, \ldots, k_d \} \), and for every

\[
(S_{i,k_1,J_i}, \ldots, S_{i,k_d,J_i}) \in S_{i,k_1,J_i} \times \ldots \times S_{i,k_d,J_i},
\]
we have $\bigcap_{j \in [d]} S_{i,k,j} \neq \emptyset$.

**Example.** Let us for instance consider the task of Fig. 2. We have $J_1 = \{1, 3\}$, and thus there are three sets in $S_1$, which are $S_{1,1,(1,3)}, S_{1,2,(1,3)}, S_{1,3,(1,3)}$. We have $Sk_{11}(O) = \{\{1, B\}\}$, and thus each of these three sets is potentially one of the four sets of sets with elements in $Sk_{11}(O)$, namely $\emptyset, \{1\}, \{B\}, \{1, B\}$, where, for the sake of simplifying the notations, we denote by $B$ the vertex $(1, B) \in Sk_{11}(O)$. However, the sets $\emptyset$ and $\{B\}$ do not satisfy $P_0$, and therefore only the two sets $\{1\}$ and $\{1, B\}$ remain. For $i \in \{2, 3\}$, we have $Sk_{11}(O) = \{(i, R), (i, G)\}$ (cf. Fig. 5). Therefore, still denoting by $X$ a vertex $(i, X) \in Sk_{11}(O)$, we get a larger collection of sets, including, e.g., $\{\{R\}, \{G\}\}$ and $\{\{R, G\}\}$. Note however, that $\{\{R\}, \{G\}\}$ can occur at most once in $S_i$ because it is not true that for any $S \in \{\{R\}, \{G\}\}$, and any $S' \in \{\{R\}, \{G\}\}$, we have $S \cap S' \neq \emptyset$. For instance, $\{R\} \cap \{G\} = \emptyset$.

**Simplices of $O'$**. A vertex-set $\{(i, (x_i, S_i)) : i \in [n]\}$ is a facet of $O'$ if, for every closed simplex $\{(i, J_i) : i \in I\} \in M$, and for every $(i, k) \in I \times I$ with $i \in J_k$ or $k \in J_i$, the following two properties hold:

**P1**: There exists $(S_{i,k,j,k, j}, S_{k,i,j,k, j}) \in S_{i,k,j,k, j} \times S_{k,i,j,k, j}$ satisfying that, for every

$$(i, y_i), (k, y_k) \in S_{i,k,j,k, j} \times S_{k,i,j,k, j},$$

there exists $\tau \in I$ such that $\{(i, x_i), (k, x_k)\} \subseteq \tau$, and $\{(i, y_i), (k, y_k)\} \in Cl(\Delta(\tau))$.

Moreover, for every $(k, J_k') \in Cl(St(i, J_i))$, and for every $(i, J_i') \in Cl(St(k, J_k))$, we have:

$S_{i,k,j,k, j} = S_{i,k,j,i, j'}$ and $S_{k,i,j,k, j} = S_{k,i,j,k, j'}$.

**P2**: $|S_{i,i,j,i}| = 1$ (i.e., a unique set in $S_{i,i,j,i}$), and, if $i \in J_k$ and $k \notin J_i$ then $S_{i,k,j,i} = S_{i,i,j,i}$.

**Example.** Let us consider again the task of Fig. 2, and let us define the following sets $S_1, S_2$, and $S_3$, where $X \in \{R, G\}$:

$S_{1,1,(1,3)} = \{\{B\}\}$, $S_{1,2,(1,3)} = \{\{B\}\}$, $S_{1,3,(1,3)} = \{\{B\}\}$

$S_{2,1,(1,2)} = \{\{X\}\}$, $S_{2,2,(1,2)} = \{\{X\}\}$, $S_{2,3,(1,2)} = \{\{X\}\}$

$S_{3,1,(2,3)} = \{\{R, G\}\}$, $S_{3,2,(2,3)} = \{\{R\}, \{G\}\}$, $S_{3,3,(2,3)} = \{\{R, G\}\}$

We claim that $\{(i, (x_i, S_i)) : i \in \{1, 2, 3\}\}$, where $x_1 = B$, $x_2 = X$, and $x_3 \in \{R, G\}$, is a facet of $O'$. First, $P_0$ is satisfied at each vertex $(i, (x_i, S_i))$, $i \in \{1, 2, 3\}$. Second, for every $i \in \{1, 2, 3\}$, the set $S_i$ satisfies $P_2$. It remains to check $P_1$. The only non-trivial case is $i = 2$ and $k = 3$. There in a unique set $\{X\}$ in $S_{2,3,(1,2)}$. Therefore, the corresponding set in $S_{3,2,(2,3)}$ must be $\{X\} = \{R, G\} \setminus \{X\}$. We pick $\tau = \{(1, B), (2, X), (3, x_3)\}$, and, indeed, $\{(2, X), (3, X)\} \subseteq \tau$ while the output of $p_2$ is equal to its input. Therefore $\{(2, X), (3, X)\} \in Cl(\Delta(\tau))$, which establishes $P_1$.

**Input-output specification.** $\Delta'$ satisfies the following:

**P3**: For every two simplices $\sigma = \{(i, x_i) : i \in I\} \in I$, and $\tau = \{(i, (x_i', S_i)) : i \in I\} \in O'$, where $I \subseteq [n]$, we set: $\tau \in \Delta'(\sigma) \iff \forall i \in I, x_i' = x_i$. 

**XX:10 On Extending Brandt’s Speedup Theorem**
Example. Still for the task of Fig. 2 the simplex \( \{(1, (B, S_1)), (2, (X, S_2)), (3, (x_3, S_3))\} \) is a valid output for the input simplex \( \{(1, B), (2, X), (3, x_3)\} \). This yields a simplicial map \( \alpha : I \to O' \) which agrees with \( \Delta' \). Therefore, \( (I, O', \Delta') \) is solvable in zero rounds. This is in agreement with our main result established in the next subsection.

2.4 General Speedup Theorem

We show that, under certain conditions on a task \( (I, O, \Delta) \), the task \( (I, O', \Delta') \) where \( O' \) is the complex defined by properties P0-2, and \( \Delta' \) is the input-output relation defined by P3, is a speedup task of \( (I, O, \Delta) \). The statements in this section are not entirely formal, as some additional properties are required for the results to hold. Nevertheless, these properties are essentially technical, and they do not impact the general message delivered by the statements below. For more details, see Sections 5.1 and 5.2.

2.4.1 From \( t \) rounds to \( t - 1 \) rounds

Lemma 3. Let \( (I, O, \Delta) \) be a task, let \( M \) be a model, let \( t \geq 1 \) be an integer, and let us assume that \( (I, O, \Delta) \) satisfies the \( (t - 1) \)-independence property w.r.t. \( M \). If \( (I, O, \Delta) \) is solvable in \( t \) rounds in \( M \), then the task \( (I, O', \Delta') \) is solvable in \( t - 1 \) rounds in \( M \).

Sketch of Proof. Let \( \delta : P^{(t)} \to O \) solving \( (I, O, \Delta) \) in \( t \) rounds in \( M \). We define \( \alpha : P^{(t-1)} \to O \). Note that, for any two closed simplices \( \varphi \) and \( \psi \) in \( St(i, J_i) \), we have \( Sk_i(\Xi(St(\varphi))) = Sk_i(\Xi(St(\psi))) \) for every simplex \( \sigma \in P^{(t-1)} \). Therefore, we can abuse notation by denoting \( Sk_i(\Xi(St(\varphi))) = Sk_i(\Xi(St(\psi))) \). For any vertex \( (i, v_i) \in P^{(t-1)} \), we let \( \alpha(i, v_i) = (i, P_i) \) with \( P_i = (x_i, S_i) \), where \( x_i \) is the input value of process \( i \) (which is present in its view \( v_i \)), and

\[
S_i = \{ S_{i,k,J_i} : ((i, J_i) \in M) \land (k \in [n]) \},
\]

where

\[
S_{i,k,J_i} = \{ \delta(Sk_i(\Xi(St(\sigma), J_i))) : (\sigma \in Sk_i(St(i, v_i))) \land (\dim(\sigma) = |\{i, k\}|-1) \}.
\]

For every \( \sigma \in Sk_i(St(i, v_i)) \) with \( \dim(\sigma) = |\{i, k\}|-1 \), the set \( S^\sigma_{i,k,J_i} = \delta(Sk_i(\Xi(St(\sigma), J_i))) \) is the set of all possible output for process \( i \) using \( \delta \) whenever the communication pattern \( J_i \) occurred at time \( t \), and the process \( k \) has its value fixed according to \( \sigma \in P^{(t-1)} \). In particular, we have \( S^\sigma_{i,k,J_i} \subseteq 2^{Sk_i(O)} \), and thus \( S_{i,k,J_i} \subseteq 2^{2^{Sk_i(O)}} \), as desired.

We show that P0 holds. For every vertex \( (i, J_i) \in M \), let us consider a set \( \{ S_{i,k,J_i} : k \in J_i \} \). By definition, for every set \( S_{i,k,J_i} \), there exists \( \sigma_k \in St(i, v_i) \) such that \( S_{i,k,J_i} = S^\sigma_{i,k,J_i} = \delta(Sk_i(\Xi(St(\sigma_k), E_i))) \). Using the \( (t-1) \)-independence property for \( k = i \in J_i \), it holds that \( \bigcup_{k \in J_i} \sigma_k \subseteq P^{(t-1)} \). There is only one candidate for the simplex corresponding to \( k = i \), and this simplex is \( \sigma_i = \{ (i, v_i) \} \). Therefore, \( \bigcup_{k \in J_i} \sigma_k = \bigcup_{k \in J_i} \sigma_k \in P^{(t-1)} \). For every \( k \in J_i \), we have \( \Xi(St(\bigcup_{k \in J_i} \sigma_k), J_i) \subseteq \Xi(St(\sigma_k), J_i) \). Therefore, \( \delta(Sk_i(\Xi(St(\bigcup_{k \in J_i} \sigma_k), J_i))) \subseteq \bigcap_{k \in J_i} S_{i,k,J_i} \). Now, \( \delta(Sk_i(\Xi(St(\bigcup_{k \in J_i} \sigma_k), J_i))) \) cannot be empty, simply because \( \bigcup_{k \in J_i} \sigma_k \in P^{(t-1)} \). Therefore, property P0 holds, that is, \( \alpha \) produces vertices of \( O' \).

To prove that \( \alpha \) solves \( (I, O', \Delta') \), it is sufficient to consider an arbitrary facet \( \rho = \{ (i, v_i) : i \in [n] \} \in P^{(t-1)} \), and its image \( \alpha(\rho) = \{ (i, (x_i, S_i)) : i \in [n] \} \), and we show that \( \alpha(\rho) \) is a facet of \( O' \) that agrees with \( \Delta' \). It is sufficient to show that both properties P1 and P2 hold as, by definition of \( \alpha \), P3 holds by construction.
First we prove that P1 holds. For every closed simplex \( \varphi = \{(i, J_i) : i \in I\} \in \mathcal{M} \), for every \((i, k) \in I \times I\) with \( j \in J_k \) or \( k \in J_i \), we consider the face \( \sigma = \{(i, v_i),(k, v_k)\} \) of \( \varphi \). Note that \( \sigma \in \text{Sk}_{i,k}(\text{St}(i, v_i)) \cap \text{Sk}_{i,k}(\text{St}(k, v_k)) \). Let us consider the sets

\[
S_{i,k,J_i}^\sigma = \delta(\text{Sk}_i(\Xi(\text{St}(\sigma), J_i))) \subseteq S_{i,k,J_i} \quad \text{and} \quad S_{k,i,J_i}^\sigma = \delta(\text{Sk}_k(\Xi(\text{St}(\sigma), J_i))) \subseteq S_{k,i,J_i}.
\]

Note that the set \( S_{i,k,J_i}^\sigma \) (resp., \( S_{k,i,J_i}^\sigma \)) is independent of \( J_k \) (resp., \( J_i \)), by construction. It can be shown, again using \((t-1)\)-independence, that P1 holds for these sets.

Finally, P2 holds, also using \((t-1)\)-independence. (See complete proof in Section 5.1.)

Note that Lemma 3 does not require local checkability, and may therefore be applied even to tasks such as consensus in \textit{WAIT-FREE}.

### 2.4.2 From \( t-1 \) rounds to \( t \) rounds

Our reciprocal of Lemma 3, which guarantees that the task \((I, O, \Delta)\) can be used for deriving a lower bound for \((I, O, \Delta)\), requires the task to satisfy a specific property, called \textit{edge-checkability}. The following definition is inspired from the notion of local checkability defined in [21] for the \textit{WAIT-FREE} model. Given a simplex \( \sigma = \{(i, x_i) : i \in I\} \), and \( J \subseteq I \), we define \( \pi_J(\sigma) = \{(i, x_i) : i \in J\} \). A task \((I, O, \Delta)\) is \textit{locally checkable} for the communication model \( \mathcal{M} \) if, for every \( \sigma \in I \) with \text{name}(\sigma) = \( I \subseteq \lceil n \rceil \), for every set \( \tau = \{(i, y_i) : i \in I \land y_i \in \text{val}(O)\} \), and for every closed simplex \( \varphi = \{(i, J_i) : i \in I\} \) of \( \mathcal{M} \), the following holds:

\[
\tau \in \Delta(\sigma) \iff \forall i \in I, \pi_J(\tau) \in \Delta(\pi_J(\sigma)).
\]

Moreover, the task \((I, O, \Delta)\) is \textit{edge-checkable} if the following holds:

\[
\tau \in \Delta(\sigma) \iff \forall i \in I, \forall k \in J_i, \pi_{\{i,k\}}(\tau) \in \Delta(\pi_{\{i,k\}}(\sigma)).
\]

Note that edge-checkability implies local checkability. For instance, renaming is edge-checkable in \textit{WAIT-FREE}, and proper coloring is edge-checkable in \textit{LOCAL}. In particular, the task depicted on Fig. 2 is edge-checkable. On the other hand, consensus is not even locally checkable in \textit{WAIT-FREE}, and the standard version of maximal independent set (MIS), where a node in the set is labeled 1, while a node not in the set is labeled 0, is locally checkable in \textit{LOCAL}, but not edge-checkable in \textit{LOCAL}. The “edge version” of MIS defined in \[\text{[8]}\] is however edge-checkable. As in \[\text{[8]}\], we assume an underlying mechanism enabling the any two processes \( i \) and \( k \) such that \( i \in J_k \) or \( k \in J_i \) to \textit{break symmetry}.

\begin{itemize}
\item \textbf{Lemma 4.} Let \((I, O, \Delta)\) be a task, let \( \mathcal{M} \) be a model, let \( t \geq 1 \) be an integer, and let \( \beta \) assume a symmetry-breaking mechanism, and that \((I, O, \Delta)\) is edge-checkable in \( \mathcal{M} \). If \((I, O', \Delta')\) is solvable in \( t-1 \) rounds in \( \mathcal{M} \) then \((I, O, \Delta)\) is solvable in \( t \) rounds in \( \mathcal{M} \).
\end{itemize}

\textbf{Sketch of Proof.} It is sufficient to show the existence of a simplicial map \( \beta : \Xi(O') \to O \) such that, for every closed simplex \( \sigma \in I \), \( \beta(\Xi(\Delta'(\sigma))) \subseteq \Delta(\sigma) \).

Let \( \tau = \{(i, x_i, S_i) : i \in I\} \in \Delta'(\sigma) \) for some closed simplex \( \sigma = \{(i, x_i) : i \in I\} \in I \). Note that \( \tau \) is a face of a facet of \( O' \), which, by definition, satisfy P1 and P2. Since \((I, O, \Delta)\) is edge-checkable, it is sufficient to prove that, for every closed simplex \( \varphi = \{(i, J_i) : i \in I\} \in \mathcal{M} \), for every \( i \in I \), process \( i \) can output a solution \( (i, y_i) \in \text{Sk}_i(O) \) such that, for every \( k \in J_i \), \( \{(i, y_i),(k, y_k)\} \in \Delta(\pi_{\{i,k\}}(\sigma)) \).

After one round of communication according to \( \varphi \), every process \( i \in I \) receives messages from every process \( k \in J_i \). Therefore every process \( i \) has access to the set \( \{(x_k, S_k) : k \in J_i\} \).

If \( k \in J_i \) does not receive from \( i \) then property P2 guarantees that \( |S_{k,i,J_k}| = 1 \). If \( k \in J_i \)
receives from \( i \) (i.e. \( i \in J_k \)) then the symmetry-breaking mechanism, and property P1 allow the process \( i \) and \( k \) to choose sets \( (S_{i,k,J_i}, S_{k,i,J_k}) \in \mathcal{S}_{i,k,J_i} \times \mathcal{S}_{k,i,J_k} \) such that, for every choice of \((y_i, y_k)\) where \((i, y_i) \in S_{i,k,J_i}\) and \((k, y_k) \in S_{k,i,J_k}\), there exists \( \tau \in \mathcal{T} \) such that

\[
\{(i, x_i), (k, x_k)\} \subseteq \tau, \text{ and } \{(i, y_i), (k, y_k)\} \in \Delta(\tau).
\]

Since \((\mathcal{I}, \mathcal{O}, \Delta)\) is edge-checkable, \(\{(i, y_i), (k, y_k)\} \in \Delta(\tau)\) implies that

\[
\{(i, y_i), (k, y_k)\} \in \Delta(\pi_{\{i,k\}}(\tau)) = \Delta(\pi_{\{i,k\}}(\sigma)).
\]

By repeating this operation for every process \( k \in J_i \), process \( i \) can output any value \( y_i \) such that,

\[
(i, y_i) \in \bigcap_{k \in J_i} S_{i,k,J_i}.
\]

Such a value \( y_i \) does exist thanks to property P0. The correctness of this algorithm is straightforward since the task is edge-checkable, and the sets \( S_{i,k,J_i} \) are precisely chosen to satisfy P1.

Using Lemmas 3 and 4 we immediately derive our main result.

\[\textbf{Theorem 5.}\] Let \((\mathcal{I}, \mathcal{O}, \Delta)\) be a task, let \( \mathcal{M} \) be a model, let \( t \geq 1 \) be an integer, and let us assume that \((\mathcal{I}, \mathcal{O}, \Delta)\) satisfies the \((t-1)\)-independence property w.r.t. \( \mathcal{M} \), and is edge-checkable in \( \mathcal{M} \). \((\mathcal{I}, \mathcal{O}, \Delta)\) is solvable in \( t \) rounds in \( \mathcal{M} \) if and only if \((\mathcal{I}, \mathcal{O}', \Delta')\) is solvable in \( t-1 \) rounds in \( \mathcal{M} \).

\section{2.5 Applications}

We illustrate the generality of Theorem 5 by examples from two radically different settings, namely shared-memory wait-free computing, and synchronous failure-free network computing.

\subsection{2.5.1 Shared-Memory Wait-Free Computing}

We consider the standard perfect renaming task in WAIT-FREE, with two processes. The two processes start with distinct identifiers in \([0, 1, 2]\) as input, and they are asked to output a distinct identifier in \([0, 1]\). We provide a new impossibility proof for perfect renaming, using Theorem 5.

\[\textbf{Corollary 6.}\] Perfect renaming in 2-process system is impossible in WAIT-FREE.

\[\textbf{Proof.}\] We start with two observations. First, for any \( t \geq 0 \), renaming satisfies \( t \)-independence w.r.t. WAIT-FREE with two processes. Second perfect renaming is edge-checkable in WAIT-FREE. Indeed, in WAIT-FREE, if every process receives identifiers that are different from its own identifier, then all identifiers are necessarily distinct. Therefore, Theorem 5 applies. Let us identify the task \((\mathcal{I}, \mathcal{O}', \Delta')\) defined by Properties P0-3 applied to perfect renaming (the input and output complexes of perfect renaming, \( \mathcal{I} \) and \( \mathcal{O} \), are displayed on Fig. 3) and the input-output specification is trivial, i.e., \( \tau \in \Delta(\sigma) \) whenever \( \text{name}(\tau) = \text{name}(\sigma) \).

In the general construction, we have \( S_j \in 2^{2^{\mathcal{O}'(\mathcal{I})}} \) for process \( j \). However, for the sake of simplifying the notations, we manipulate sets in \( 2^{\mathcal{O}'(\mathcal{I})} = 2^{2^{\mathcal{O}'(\mathcal{O})}} \). Any set \( S \in 2^{\mathcal{O}'(\mathcal{O})} \) must be one of the following three sets: \( \emptyset = \{0\} \), \( 1 = \{1\} \), and \( X = \{0, 1\} \). Indeed, the empty set \( \emptyset \neq \sigma \) does not satisfy P0. We denote by \( p_i \) and \( p_k \) the two processes in the systems (instead of \( p_1 \) and \( p_2 \), for avoiding confusion between process indexes and input and output values).
Let us consider a facet \( \{ (\ell, (x_\ell, S_\ell)) : \ell \in \{ i, k \} \} \subseteq \mathcal{O}' \). By definition, for any \( \ell \in \{ i, k \} \), we have

\[
S_\ell = \{ S_{\ell', J_k} : (\ell, J_\ell) \in \mathcal{M} \land \ell' \in \{ i, k \} \}.
\]

Let us focus on \( S_{i,k,J} \) and \( S_{k,i,J} \) for \( J = \{ i, k \} \). Recall that any \( S \subseteq S_{i,k,J} \) (resp., \( S_{k,i,J} \)) is non-empty thanks to the universal quantifier in \( P_0 \). Moreover, \( S_{i,k,J} \) (resp., \( S_{k,i,J} \)) is itself non-empty thanks to the existential quantifier in \( P_1 \). (These two facts actually hold for any set in \( S_i \) or \( S_k \).) By Property \( P_1 \), there exists \((S_{i,J}, S_{k,J}) \subseteq S_{i,k,J} \times S_{k,i,J} \) such that, for every \((y_i, y_k) \in S_{i,J} \times S_{k,J} \),

\[
\{(i, y_i), (k, y_k)\} \in \Delta(\{(i, x_i), (k, x_k)\}) = \mathcal{O}.
\]

We necessarily have \( S_{i,J} \neq X \) because, for any \( y \in \{ 0, 1 \} \), either \( (i, 0), (k, y) \) \( \notin \mathcal{O} \) or \( (i, 1), (k, y) \) \( \notin \mathcal{O} \), and therefore it is not possible that both sets are simplices of \( \mathcal{O} \). By the same arguments, we also have \( S_{k,J} \neq X \). It follows that \( S_{i,k,J} \) and \( S_{k,i,J} \) can only take three possible values: \( \mathcal{O} = \{ 0 \}, I = \{ 1 \} \) and \( X = \{ 0, 1 \} \).

By the same arguments applied on the facet \( \{(i, S_{i,k,J}). (k, S_{k,i,J})\} \), we get that \( S_{k,i,J} \subseteq \mathcal{O} \) can also only take its values in \( \{ 0, I, X \} \). By Property \( P_2 \), it must be the case that \( S_{k,i,J} = S_{k,k,J} \). On the other hand, \( |S_{k,k,J}| = 1 \) implies that \( S_{k,i,J} \in \{ 0, I \} \).

Symmetrically the same holds for process \( i \), that is, \( S_{i,k,i,J} \in \{ 0, I \} \).

We now show that, necessarily, \( S_{i,k,J} \neq S_{k,i,J} \). Let us consider the two sets \( S_{i,J} \subseteq S_{i,k,J} \) and \( S_{k,J} \subseteq S_{i,k,J} \) such that \( (S_{i,J}, S_{k,J}) \) and \( (S_{i,k,J}, S_{i,J}) \) both satisfy \( P_1 \). Since \( (S_{i,J}, S_{k,J}) \) satisfy \( P_1 \), and since \( (S_{i,J}, S_{k,J}) \subseteq \{ 0, I \} \), we have \( S_{i,J} \neq S_{k,J} \). It follows that \( S_{k,J} = S_{i,J} \), and therefore \( S_{i,J} = S_{i,J} \). Similarly, we have \( S_{i,J} \neq S_{k,i,J} \), and therefore \( S_{k,J} = S_{k,k,J} \). As a consequence, \( S_{i,J} \neq S_{k,k,J} \), and thus \( S_{i,k,J} \neq S_{k,i,J} \), as claimed.

Overall, we have shown that \( (S_{i,k,J}, S_{k,i,J}) \subseteq \{ 0, I, (I, 0) \} \). Therefore, by replacing \( \mathcal{O} \) by \( 0 \), and \( I \) by \( 1 \), there is a one-to-one correspondence between the partial outputs \( (S_{i,k,J}, S_{k,i,J}) \) (resp., \( (S_{i,J}, S_{k,J}) \)) for \( (I, O', \Delta') \), and valid outputs for perfect renaming. It follows that if there is an algorithm for solving the task \( (I, O', \Delta') \) defined by Properties \( P_0-P_3 \) applied to perfect renaming, then, in particular, this algorithm also solves perfect renaming. Therefore, thanks to Theorem 6, we get that, for every \( t \geq 1 \), if perfect renaming is solvable in \( t \) rounds in \( \text{WAIT-FREE} \), then perfect renaming is solvable in \( t - 1 \) rounds in \( \text{WAIT-FREE} \). Since perfect renaming is not solvable in zero rounds, we conclude that perfect renaming is not solvable in \( \text{WAIT-FREE} \). ▶

**Remark.** The proof of Corollary 6 illustrates a quite interesting case, where solving the speedup task \( (I, O', \Delta') \) obtained using \( P_0-3 \) includes solving the original task \( (I, O, \Delta) \). In this case, Theorem 4 is not necessary, and Lemma 3 suffices for establishing the impossibility of solving the task \( (I, O, \Delta) \). That is, the edge-checkability condition is not required, and solely local-independence is required. An interesting application is 2-process consensus, which is not edge-checkable in \( \text{WAIT-FREE} \). Nevertheless, the speedup task of consensus obtained using Property \( P_0-3 \) happens to include consensus itself, by the same type of arguments as in the proof of Corollary 6. Impossibility of consensus therefore directly follows from Lemma 3.

**Corollary 7.** Consensus in 2-process system is impossible in \( \text{WAIT-FREE} \).

### 2.5.2 Synchronous failure-free network computing

As mentioned before, several models can satisfy the conditions in the statement of Theorem 4 beyond \text{LOCAL}. This is, for instance, the case for some dynamic graph models, \text{DYN}(\mathcal{F}),
where, at each round, one of the graphs in $F$ is chosen to be the underlying communication graph. This is also the case for some hypergraph models, $H\text{-LOCAL}(G)$, which is the natural extension of LOCAL to hypergraphs $G$ ($H$ stands for hypergraph). It can be shown that, in $H\text{-LOCAL}$, edge-checkability is essentially equivalent, up to $\pm 1$ rounds, to local checkability (see Lemma 20). The following result is a direct consequence of our generalised version of Theorem 5 (see Theorem 19). Interestingly, as communications produced by hypergraphs could be viewed as communications on a graph where hyperedges are transformed into cliques, the result below also shows that the “large girth property” is, to some extend, not necessary for applying Theorem 5. For instance, Theorem 5 could be applied to communication graphs such as the one represented in Figure 6. The proof of the following can be found in Section 5.3.

\[\text{Corollary 8.} \] Let $H$ be a hypergraph on $n$ nodes, let $\Pi = (I, O, \Delta)$ be a task for $n$ processes, and let $t \geq 1$. Let us assume that $\Pi$ satisfies the $r$-independence property w.r.t. $H\text{-LOCAL}(H)$ for every $r \in \{0, \ldots, t - 1\}$, and that $\Pi$ is edge-checkable in $H\text{-LOCAL}(H)$. Let us assume the existence of a symmetry breaking mechanism among the processes in each hyperedge of $H$. $\Pi$ is solvable in $t$ rounds in $H\text{-LOCAL}(H)$ if and only if $\Pi^{(t)} = (I, O^{(t)}, \Delta^{(t)})$ is solvable in zero rounds in $H\text{-LOCAL}(H)$, where $\Pi^{(t)}$ is the task defined by iterating $t$ times the construction defined by properties P0-3.

### 2.6 Technical Summary

Brandt’s speedup theorem holds thanks to an operator $\Phi$ which, given any task $\Pi = (I, O, \Delta)$, constructs a task $\Phi(\Pi) = (I, O', \Delta')$ such that, for every $t \geq 1$, if $\Pi$ is $t$-independent w.r.t. the LOCAL model, and if $\Pi$ is edge-checkable in the LOCAL model then

$\Pi$ is solvable in $t$ rounds in LOCAL $\iff$ $\Phi(\Pi)$ is solvable in $t - 1$ rounds in LOCAL.

We have extended Brandt’s operator $\Phi$ to an operator $\Phi^*$ defined by Properties P0-3, which applies to any round-based model $M$ supporting full-information protocols. We have extended the notion of local independence and edge-checkability to these models, which allows us to extend the equivalence above to all such communication model $M$, using $\Phi^*$ instead of $\Phi$. Interestingly, the operator $\Phi^*$ is applicable even to asynchronous models such as WAIT-FREE, and allows us to provide new impossibility proofs for consensus and renaming in 2-process systems. Defining an operator $\Phi_{WF}$ transforming any task $\Pi = (I, O, \Delta)$ into a task $\Phi_{WF}(\Pi) = (I, O', \Delta')$ such that, under some conditions, for every $t \geq 1$ and $n \geq 2$, $\Pi$ is solvable in $t$ rounds in WAIT-FREE if and only if $\Phi_{WF}(\Pi)$ is solvable in $t - 1$ rounds in WAIT-FREE. Our operator $\Phi^*$ applies to WAIT-FREE for $n = 2$. Extending $\Phi^*$ to some $\Phi_{WF}$ for arbitrary $n$ requires to overcome the fact that WAIT-FREE does not satisfy local independence for $n > 2$. 

---

*Figure 6* A graph induced by a linear hypertree
The remaining of the paper formalizes and generalizes the concepts and ideas provided in this section.

3 A General Model of Communication

This section describes our framework. We assume that the reader is familiar with the basic concepts of combinatorial topology applied to distributed computing, and in particular with the notion of task \((I, O, \Delta)\). The reader unfamiliar with these concepts may refer to Appendix A. Our framework encapsulates several standard communication media, from asynchronous crash-prone shared-memory computing to synchronous failure-free network computing, such as:

- **WAIT-FREE**\((n)\) refers to the asynchronous shared-memory model involving \(n\) crash-prone processes interacting via iterated immediate snapshots \([3]\). Note that this model is computationally equivalent to the model with atomic read/write operations \([27]\). \(f\)-**RESILIENT**\((n)\) refers to the same model as WAIT-FREE\((n)\), except that at most \(f\) processes can crash, where \(1 \leq f \leq n - 1\) \([3]\).

- **LOCAL**\((G)\) refers to the synchronous failure-free message-passing model in networks \([36]\), i.e., processes are nodes of the (undirected) graph \(G\), computation and communication proceed in lockstep, and a message is exchanged along each edge of \(G\) between neighboring processes at every time step. This model can trivially be extended to **H-LOCAL**\((H)\), where \(H\) is a hypergraph.

- **DYN**\((F)\) refers to the dynamic network model, which is the same as LOCAL except that the communication graph may evolve with time \([12,16]\). Specifically, given a family \(F\) of \(n\)-node graphs on the same set of vertices, the communications occurring at any step are performed along the edges of one of the graphs in \(F\). Again, this model can trivially be extended to **H-DYN**\((F)\) where \(F\) is a family of hypergraphs.

All these models share properties that are essential to our framework. First, they all include a notion of **round**, either explicitly like in LOCAL and DYN, or implicitly like in WAIT-FREE and f-RESILIENT. Second, they all support full information protocols, that is, whenever a process communicates, whether it be by sending/receiving messages to/from neighbors, or writing/reading in a shared memory, it communicates its entire history since the beginning of its execution. This assumption enables the design of robust lower bounds, which still hold if the communication medium restricts the communication power somehow. Last but not least, none of these models place limitations on the individual computing power of each process, which enables the design of unconditional lower bounds or impossibility results. Note that none of these models refer to IDs. The fact that nodes may or may not be provided with IDs is, in our framework, not a property of the communication model, but a property of the tasks to be solved in this model. This is formally specified below.

**Names and Identifiers.** Identifiers can be viewed as forming a special type of input values, for which it is assumed that every two different processes are assigned different IDs. We stress the fact that the ID of a process must not be confused with its name. The former is a value given to the process as input, while the latter is external, whose sole purpose it to refer to the process. In this paper, we always assume that processes are not aware of their names, and processes may or may not be granted with IDs as part of their inputs. For \(i \in [n]\), process \(i\) is denoted by \(p_i\), and its identifier (if any) by \(\text{id}(p_i)\). It is systematically the case that \(\text{name}(p_i) = i\), while \(\text{id}(p_i)\) is some arbitrary value taken in some finite set of integers. When IDs are assigned to the processes, the input value of \(p_i\) is therefore a pair \((x, y)\) where
The semantics of such a vertex is that \( x \) is the ID of \( p_i \), and \( y \) is the input label of \( p_i \). Hence, a vertex of the input complex is of the form \( (i,s) = (\text{name}(p_i), \text{val}(p_i)) = (\text{name}(p_i), (\text{id}(p_i), \text{label}(p_i))) \). A task \((\mathcal{I}, \mathcal{O}, \Delta)\) does not necessarily involve IDs in its specification, e.g., consensus. However, given a tasks \((\mathcal{I}, \mathcal{O}, \Delta)\) with no IDs, solving \((\mathcal{I}, \mathcal{O}, \Delta)\) with IDs in \([1, N]\) is merely the task \((\mathcal{I}', \mathcal{O}, \Delta')\) where \( \{(i,(x_i, v_i)) : i \in I\} \) is in \( \mathcal{I}' \) if (1) \( \{(i, v_i) : i \in I\} \) is in \( \mathcal{I} \), and, (2) for every \( i \in I \) \( x_i \in [1,N] \) with \( x_i \neq x_j \) for every \( j \in I \setminus \{i\} \). Moreover, for every \( \sigma' = \{(i, (x_i, v_i)) : i \in I\} \) in \( \mathcal{I}' \) and \( \tau \in \mathcal{O} \), we set \( \tau \in \Delta'(\sigma') \) whenever \( \tau \in \Delta(\sigma) \) where \( \sigma = \{(i, v_i) : i \in I\} \in \mathcal{I} \).

**Communication Models.** We now describe a way to encapsulate various communication models in a single general framework. Every communication medium is modeled as a complex \( \mathcal{M} \). To describe this complex, let \( \mathcal{S}_n \) be the \((n-1)\)-dimensional chromatic pseudo-sphere induced by \( 2^{|n|} \) (cf. Appendix A.1). A vertex of \( \mathcal{S}_n \) is a pair \((i,E)\) with \( i \in [n], \) and \( E \subseteq 2^n \).

The semantics of such a vertex is that \( p_i \) receives information from all \( p_j, j \in E, \) for every \( e \in E \). Given a vertex \((i,E)\) of \( \mathcal{S}_n \), each element \( e \in E \) is called a channel. In fact, we will consider only vertices \((i,E)\) where \( \{i\} \in E \), that is, we systematically assume that a process has a (private) channel to itself, a.k.a. a “self-loop”. Also, w.l.o.g., for simplifying and unifying the presentation, we will consider only vertices \((i,E)\) where, for every \( e \in E, i \in E \).

**Definition 9.** A communication model is a pure \((n-1)\)-dimensional sub-complex \( \mathcal{M} \) of \( \mathcal{S}_n \) such that, for every vertex \((i,E)\in \mathcal{M}, \{i\} \in E, \) and, for every \( e \in E, i \in E \).

In many classical models, every channel has cardinality 2 (putting aside the self-loops). Given a vertex \((i,E)\), and a channel \( e = \{i,j\} \in E \), the semantic of this channel is that \( p_i \) receives information from \( p_j \) via the channel \( e \). Whenever all channels have cardinality \( \leq 2 \), the vertices of \( \mathcal{M} \) can merely be represented as pairs \((i,E)\) where \( E \subseteq [n] \), and \( i \in E \). In that case, an element \( j \in E \) represents a channel \( \{i,j\} \). It is however worth considering the case where channels have larger cardinalities, as it actually helps in the statement of our results, and it makes these results directly applicable to models involving multiparty communications, e.g., involving hypergraphs (a channel is then merely a hyperedge).

The semantics of a simplex \( \{(i,E_i) : i \in I\}, I \subseteq [n], \) of \( \mathcal{M} \) is that the model allows an instance of communication in which, during a same round, every process \( i \in I \) receives a message from all processes in the channels belonging to \( E_i \). Here are a few examples, where the first three deal with channels involving two processes (therefore the sets \( E_i \) are merely subsets of \([n]\)), while the fourth one involves potentially larger channels.

- **WAIT-FREE**\((n)\): for every non-empty \( I \subseteq [n], \) a set \( \{(i,E_i) : i \in I\} \) is a simplex of \( \mathcal{M} \) if, for every \( i,j \in I \), we have \( E_i \subseteq [n], \) \( E_i \subseteq E_j \lor E_j \subseteq E_i \), and we have \( E_j \subseteq E_i \) whenever \( j \in E_i \). \( \mathcal{M} \) is actually isomorphic to the chromatic subdivision of the complete complex with vertex set \( V = \{(i,\bot) : i \in [n]\} \) (see [28]).
- **f-RESILIENT**\((n)\): for every non-empty \( I \subseteq [n], \) a set \( \{(i,E_i) : i \in I\} \) is a simplex of \( \mathcal{M} \) if, for every \( i,j \in I \), we have \( E_i \subseteq [n], \) \( E_i \subseteq E_j \lor E_j \subseteq E_i \), \( E_j \subseteq E_i \) whenever \( j \in E_i \), and \( |E_i| \geq n-f \). Indeed, since at most \( f \) processes can crash, every process can wait for at least \( n-f-1 \) other processes before proceeding to the next round.
- **LOCAL**\((G)\): given an \( n \)-node graph \( G \) with nodes labeled from 1 to \( n \), for every non-empty \( I \subseteq [n], \) a set \( \{(i,E_i) : i \in I\} \) is a simplex of \( \mathcal{M} \) if, for every \( i \in I \), \( E_i = N_G[i] \) where \( N_G[i] \) denotes the closed neighborhood of node \( i \) in \( G \). In **DYN**\((F)\), given a family \( F \) of \( n \)-node graphs with nodes labeled from 1 to \( n \), for every non-empty \( I \subseteq [n], \) a set \( \{(i,E_i) : i \in I\} \) is a simplex of \( \mathcal{M} \) if there exists \( G \in F \) such that, for every \( i \in I \), \( E_i = N_G[i] \).
- **H-LOCAL**\((H)\): given an \( n \)-node hypergraph \( H \) with nodes labeled from 1 to \( n \), for every non-empty \( I \subseteq [n], \) a set \( \{(i,E_i) : i \in I\} \) is a simplex of \( \mathcal{M} \) if, for every \( i \in I \), \( E_i = E_H(i) \)
On Extending Brandt’s Speedup Theorem

where $E_H(i)$ denotes the set of hyperedges of $H$ containing node $i$. For the sake of technical uniformity, we assume, w.l.o.g., that $\{i\} \in E_H(i)$.

Some models $\mathcal{M}$ are non-deterministic, in the sense that, given a set $I \subseteq [n]$ of processes, there are more than one simplices $\sigma \in \mathcal{M}$ with $\text{name}(\sigma) = I$. This is the case for \textsc{wait-free}, \textsc{f-resilient}, and \textsc{dyn}. This reflects the fact that the pattern of communication may differ at each round, whether it be because the processes are asynchronous, because some processes have crashed, or because the communication network evolves with time. Instead, \textsc{local} and \textsc{h-local} are deterministic in the sense that the communication pattern performed at each round is identical through time. Beyond the standard models captured by our formalism, the formalism is flexible enough to capture a vast class of other models, including the following asynchronous wait-free variant of \textsc{local}.

- \textsc{wf-local}(G) denotes the model $\mathcal{M}$ in which, given an $n$-node graph $G$ with nodes labeled from 1 to $n$, for every non-empty $I \subseteq [n]$, a set $\{(i, E_i) : i \in I\}$ is a simplex of $\mathcal{M}$ if, for every $i \in I$, we have $E_i \subseteq N_G[i]$, and, for every $U \subseteq I$, if the subgraph of $G$ induced by $U$ is a clique, then for every $i, j \in U$, $(E_i \cap U \subseteq E_i \cup E_j \cap U \subseteq E_i \cap U)$, and $E_i \cap U \subseteq E_j \cup E_j$ whenever $j \in E_i$. (In particular, for every edge $\{i, j\} \in E(G)$, $j \in E_i$ or $i \in E_j$, or both).

Open and Closed Simplices. There are two types of simplices in $\mathcal{M}$. A closed simplex is a simplex $\{(i, E_i) : i \in I\}$ such that the set of processes in the union of all channels is $I$, i.e., $\bigcup_{i \in I} \bigcup_{e \in E_i} e = I$. A simplex that is not closed is called open. Closed simplices play an important role as they can be used to model various scenarios in which the processes in $I$ are disconnected from the other processes, whether it be because the latter crashed, or because the processes in $I$ are forming a connected component of a disconnected network. In \textsc{wait-free}(n), for every $I \subseteq [n]$, there are simplices $\sigma \in \mathcal{M}$ with $\text{name}(\sigma) = I$ that are closed, and there are simplices $\sigma \in \mathcal{M}$ with $\text{name}(\sigma) = I$ that are open. Instead, in \textsc{local} and \textsc{dyn}, as long as the networks are connected, only the facets of $\mathcal{M}$ are closed, and all the lower dimensional simplices are open. Similarly, in \textsc{f-resilient}(n), any simplex $\sigma$ with $|\text{name}(\sigma)| < n - f$ is necessarily open.

Local Encoding of the processes and channels. In the presence of IDs assigned to the processes, a process $i$ can trivially identify the other processes from which it receives information thanks to their IDs. In absence of IDs, and/or in the presence of large channels, we assume a mechanism baring similarities with the port-numbers in anonymous variants of \textsc{local} [29]. Specifically, for every process $i$, each channel $e$ incident to $i$ is uniquely identified by $p_i$, and every process $j \in e$ is also unambiguously identified by $p_i$, that is, if $j \in e \cap e'$ for two channels $e$ and $e'$ incident to $i$, process $i$ correctly identifies the information received from $j$ as information received from a same process. Moreover, all processes in a same channel $e$ identify $e$ with the same name.

Communication Map and Protocol Complexes. We now describe the evolution of the system along with time when a full-information protocol is executed for solving a task. To every communication model $\mathcal{M}$ corresponds a communication map $\Xi$ that applies to any chromatic complex $K$, defined as follows. Let $\sigma = \{(i, v_i) : i \in I\} \in K$, where $I \subseteq [n]$, and let us assume that there exists a closed simplex $\varphi = \{(i, E_i) : i \in I\}$ in $\mathcal{M}$. We set

$$\Xi(\sigma, \varphi) = \{\{i, \{v_j : j \in e\} : e \in E_i\} : i \in I\}. $$
Note that, for every \( i \in I, \) and every \( e \in E_i, \) \( \{v_j : j \in e\} \) is a multiset, and so is \( \{\{v_j : j \in e\} : e \in E_i\}. \) The elements of both multisets are indexed locally by process \( i, \) thanks to the local identification of the channels and of the other processes. For every \( \sigma \in K, \) we define \( \Xi(\sigma) = \{\Xi(\sigma, \varphi) : (\varphi \in M) \land (\text{name}(\varphi) = \text{name}(\sigma)) \land (\varphi \text{ is closed})\}. \) The communication map \( \Xi \) merely reflects the fact that if the processes in \( \sigma \) interact among themselves according to \( \varphi \in M, \) then they will end up in the state \( \Xi(\sigma, \varphi) \) in \( \Xi(\sigma). \) Note that if all simplices \( \varphi \in M \) with \( \text{name}(\varphi) = \text{name}(\sigma) \) are open, then \( \Xi(\sigma) = \emptyset. \) An empty \( \Xi(\sigma) \) reflects the fact that there are no communication patterns for which the processes in \( \text{name}(\sigma) \) communicate solely among themselves. We say that a simplex \( \sigma \in K \) is closed in model \( M \) if there exists a closed simplex \( \varphi \in M \) with \( \text{name}(\varphi) = \text{name}(\sigma). \) The simplex \( \sigma \) is open otherwise.

\[ \text{Definition 10.} \text{ Given a communication model } M, \text{ and given a pure } (n - 1)\text{-dimensional chromatic complex } K, \text{ we define } \Xi(K) \text{ as the closure of the images by } \Xi \text{ of all the closed simplices of } K. \text{ In other words, } \Xi(K) = \bigcup_{\sigma \in K} \text{ closed } Cl(\Xi(\sigma)). \]

Note that, by definition of the closure operator, \( \Xi(K) \) is a complex. Moreover, this complex is pure, with dimension \( n - 1. \) It is precisely the complex representing all possible states of the system after one round of communication starting from the state complex \( K. \) The communication map \( \Xi \) induced by a model \( M \) enables to specify the evolution of the system along with the course of an execution starting from any initial state.

\[ \text{Definition 11.} \text{ Given the input complex } I \text{ of some task } (I, O, \Delta), \text{ the protocol complex at time } t \geq 0, \text{ denoted by } \mathcal{P}(t), \text{ is defined as } \mathcal{P}(0) = I, \text{ and, for every } t > 0, \mathcal{P}(t) = \Xi(\mathcal{P}(t-1)). \]

By definition, a vertex of the protocol complex \( \mathcal{P}(t) \) is a pair \((i, w)\) where \( w \) is a possible view of the system as perceived by process \( i \) at time \( t \). This view depends on the history experienced by process \( i \) during \( t \) rounds of communication, and is very much depending on the communication model. For instance, in \( \text{LOCAL}(G) \), the view \( w \) is a ball of radius \( t \) in the graph \( G, \) whose nodes are labeled by their input values (and their IDs if the task at hand assumes identifiers). In \( \text{WAIT-FREE}, \) \( w \) results from the sequence of \( t \) immediate-snapshot instructions performed at the successive levels \( 1 \) to \( t \) of the shared memory, whose values depend on the interleavings of these instructions performed asynchronously by all the processes. By definition, a set \( \{((i, w_i) : i \in I\} \) of vertices of \( \mathcal{P}(t), \) with \( I \subseteq [n] \), forms a simplex of \( \mathcal{P}(t) \) if the views \( w_i, i \in I, \) are mutually compatible, i.e., there is a sequence of \( t \) communication rounds leading every process \( i \in I \) to acquire the view \( w_i. \)

**Computation as Simplicial Maps.** In the context of full-information protocols, the design of an algorithm boils down to computing an output at every process after a given number \( t \geq 0 \) of communication rounds. That is, a \( t \)-round algorithm consists of (1) communicating for \( t \) rounds, and (2) computing an output at each process. A \( t \)-round algorithm for a task \((I, O, \Delta)\) is therefore merely a chromatic (i.e., name-preserving) function \( \delta : V(\mathcal{P}(t)) \rightarrow V(O) \) mapping every vertex \((i, s) \in V(\mathcal{P}(t)) \) to some vertex \((i, v) \in V(O). \) The semantic of this mapping is that process \( i \) in state \( s \) at round \( t \) outputs the value \( v. \) For the algorithm to be correct, it must be the case that \( \delta \) is simplicial, i.e., it maps simplices to simplices. Indeed any possible global state of the system at time \( t \) must be mapped to some legal global output state. Moreover, \( \delta \) must agree with the specification \( \Delta \) of the task at hand, that is, for every closed simplex \( \sigma \in \mathcal{I}, \) it must be the case that \( \delta(\Xi(\sigma)) \subseteq \Delta(\sigma). \) This guarantees that, for every closed input state \( \sigma \in \mathcal{I}, \) which may evolve in any of the states \( \tau_1, \ldots, \tau_k \) of \( \mathcal{P}(t) \) after \( t \) rounds (i.e., after \( t \) applications of \( \Xi), \) the output \( \delta(\tau_i) \) of each of these states is legal w.r.t.
On Extending Brandt’s Speedup Theorem

the specification $\Delta$ of the task, stating that the output state $\delta(\tau)$ must be one of the states listed in $\Delta(\sigma)$. Note that if $\sigma$ is closed, then so are all simplices in $\Xi(\sigma)$, and therefore $\Xi^i(\sigma) \neq \emptyset$. The inclusion $\delta(\Xi^i(\sigma)) \subseteq \Delta(\sigma)$ is not enforced for open simplices $\sigma$. This is because there are no executions in $M$ that let the processes in $\sigma$ solely interacting among themselves. Finally, the map $\delta$ must be name-independent, that is, for every two vertices $(i, w)$ and $(j, w)$ of $\mathcal{P}(t)$, one must have $\delta(i, w) = (i, x)$ and $\delta(j, w) = (j, x)$, for the same output value $x$. This is because the name $i$ of process $i$ is not part of its input. The identifier $\text{id}(p_i)$ may however be part of $p_i$’s input. Obviously, a setting assuming that $\text{id}(p_i) = i$ for every $i \in [n]$, may equivalently be viewed as assuming no IDs, and then disregarding the constraint of name-independence by allowing the outcomes of $\delta$ to depend on the names.

**Computability** The following theorem (whose proof can be found in Appendix B, for the sake of completeness) fully characterizes the ability to solve a task in a given number of rounds. It can be summarized as “the diagram in Figure 1 commutes”. Its statement is a straightforward generalization of similar statements in [27].

**Theorem 12.** Let $M$ be a communication model, let $\Xi$ be the associated communication map, and let $(I, O, \Delta)$ be a task. For every $t \geq 0$, there exists a $t$-round algorithm solving $(I, O, \Delta)$ in model $M$ if and only if there exists a chromatic name-independent simplicial map $\delta : \mathcal{P}(t) \rightarrow O$ such that, for every closed simplex $\sigma \in I$, $\delta(\Xi(\sigma)) \subseteq \Delta(\sigma)$.

4 Speedup Theorem

This section presents another illustration of the flexibility of our model, by establishing a result generalizing to other models the speedup theorem by Brandt [8] stated for LOCAL. Given a task $T = (I, O, \Delta)$ Brandt’s speedup theorem enables to automatically construct a task $T' = (I, O', \Delta')$ such that $T$ is solvable in $t$ rounds in LOCAL($G$) if and only if $T'$ is solvable in $t - 1$ rounds in LOCAL($G$). Although the theorem holds for specific graphs $G$ only, and under some conditions to be satisfied by the task $T$, it provides a powerful tool for the design of lower bounds. For extending this theorem to models beyond LOCAL, we define the notion of speedup in general. Let $M$ be a communication model, and let $\Xi$ be its associated communication map.

**Definition 13.** Let $T = (I, O, \Delta)$ be a task. A task $T' = (I, O', \Delta')$ is a speedup of $T$ for $M$ if the following two conditions hold:

1. for every $t \geq 1$, if there exists a $t$-round algorithm solving $T$ in $M$, then there exists a simplicial map $\alpha : \mathcal{P}^{(t-1)} \rightarrow O'$ such that, for every closed simplex $\sigma \in I$, $\alpha(\Xi^{t-1}(\sigma)) \subseteq \Delta'(\sigma)$;
2. there exists a simplicial map $\beta : \Xi(O') \rightarrow O$ such that, for every closed simplex $\sigma \in I$, $\beta(\Xi(\Delta'(\sigma))) \subseteq \Delta(\sigma)$.

Figure 4 provides a graphical representation of Definition 13. The first condition expresses the fact that the task $T'$ is solvable in $t - 1$ rounds whenever the original task $T$ is solvable in $t$ rounds. The second condition essentially expresses the fact that the task $T$ is solvable in a single round whenever the processes are given as input a solution for $T'$. The second condition therefore guarantees that the round-complexity of task $T'$ does not decrease too much compared to the complexity of task $T$.

**Lemma 14.** If a task $T'$ is a speedup of a task $T$ in $M$, then, for every $t \geq 1$, $T$ is solvable in at most $t$ rounds in $M$ if and only if $T'$ is solvable in at most $t - 1$ rounds in $M$. 
Proof. If $\mathcal{T}$ is solvable in at most $t$ rounds then the first condition of Definition 13 guarantees the existence of a simplicial map $\alpha : \mathcal{P}^{(t-1)} \to \mathcal{O}'$ that agrees with $\Delta'$. Thanks to Theorem 12 this guarantees that $\mathcal{T}'$ is solvable in $t - 1$ rounds.

Conversely, let us assume that $\mathcal{T}'$ is solvable in at most $t - 1$ rounds. A $t$-round algorithm for $\mathcal{T}$ proceeds in two phases, as follows. During the first phase, $t - 1$ communication rounds are performed, and a solution for $\mathcal{T}'$ is computed at each process. During the second phase, a single round of communications is performed, during which the processes exchange their outputs for $\mathcal{T}'$. After this final round, every process outputs the value resulting from the application of $\beta$ to the collection of outputs in $\text{val}(\mathcal{O}')$ collected during the second phase. Let $\sigma \in \mathcal{I}$ be a closed simplex. By Theorem 12, the first phase results in a global state $\tau \in \Delta'$. Let $\tau' \in \Xi(\tau)$. The second condition of Definition 13 guarantees that $\beta(\tau') \in \Delta(\sigma)$. Therefore, our $t$-round algorithm allows the processes to output a solution for $\mathcal{T}$ that agrees with the input $\sigma$.

5 Generalized Brandt’s Speedup Mechanism

We now introduce a general approach susceptible to construct a speedup task $\mathcal{T}'$ of any given task $\mathcal{T} = (\mathcal{I}, \mathcal{O}, \Delta)$, under any model $\mathcal{M}$. Let $\delta : \mathcal{P}^{(t)} \to \mathcal{O}$ be a simplicial map that agrees with $\Delta$. There is a natural candidate $\mathcal{T}' = (\mathcal{I}, \mathcal{O}', \Delta')$ that may be a speedup of $\mathcal{T}$. Given a vertex $(i, v) \in \mathcal{P}^{(t-1)}$, recall that we denote by $\text{St}(i, v)$ the actual closure $\text{Cl}(\text{St}(i, v))$ of the star $\text{St}(i, v)$. Hence, $\text{St}(i, v)$ is a complex (while $\text{St}(i, v)$ is usually the set of all simplices of $\mathcal{P}^{(t-1)}$ containing $(i, v)$). Let us set $\alpha(i, v) = \left\{ (i, \delta(\Xi(\text{St}(i, v)))) \right\}$ for every $(i, v) \in \mathcal{P}^{(t-1)}$. We have that $\Xi(\text{St}(i, v))$ is a complex, that is pure, and of dimension $n - 1$. This complex is actually a sub-complex of $\mathcal{P}^{(t)}$. The simplicial map $\delta$ maps this sub-complex to a sub-complex of $\mathcal{O}$. In other words, the image of a vertex $(i, v) \in \mathcal{P}^{(t-1)}$ is a pair $(i, K)$ where $K$ is a pure $(n - 1)$-dimensional sub-complex of $\mathcal{O}$. Therefore, a good candidate for $\mathcal{O}'$ is the chromatic simplicial complex on vertex set $V(\mathcal{O}') = \{ \alpha(i, v) : (i, v) \in \mathcal{P}^{(t-1)} \}$, such that a non-empty set $\{ (i, K_i) : i \in I \}$ of vertices of $\mathcal{O}'$, with $I \subseteq [n]$, is a simplex of $\mathcal{O}'$ if there exists a simplex $\{ (i, v_i) : i \in I \} \in \mathcal{P}^{(t-1)}$ such that, for every $i \in I$, $K_i = \alpha(i, v_i)$. By construction, the map $\alpha : \mathcal{P}^{(t-1)} \to \mathcal{O}'$ is simplicial, which is the first requirement of Definition 13. However, it is uneasy to push this approach further, for two reasons. First, $\mathcal{O}'$ depends on $\delta$, and, second, it is not clear which conditions need to be satisfied for guaranteeing the existence of $\beta : \Xi(\mathcal{O}') \to \mathcal{O}$. Yet, we were inspired by this approach for our generalization of Brandt’s speedup theorem. In essence, the approach followed in the proof of Brandt’s speedup theorem consists of finding a clever way to split the complex $\delta(\Xi(\text{St}(i, v)))$ into a collection of sets of output values, one for each edge incident to $p_i$ in the graph $G$ of LOCAL$(G)$. We shall follow the same approach, for each channel incident to process $i$ in $\mathcal{M}$.

First, we present a speedup construction satisfying the first condition of Definition 13 only. Obviously, satisfying that first condition is straightforward, by taking any task $\mathcal{T}' = (\mathcal{I}, \mathcal{O}', \Delta')$ solvable in zero rounds (e.g., pick $\mathcal{O}' = \text{Cl}(\{ (i, \bot) : i \in [n] \})$) with $\Delta'(\sigma) = \{ (i, \bot) : i \in \text{name}(\sigma) \}$ for any $\sigma \in \mathcal{I}$). Our construction is non-trivial in the sense that, under additional assumptions, it also satisfies the second condition of Definition 13. In absence of these assumptions, only the first condition is guaranteed to hold, and thus we call that result the weak speedup lemma. Note that, depending on the context, such a lemma may potentially yield a task $\mathcal{T}'$ whose complexity is smaller than $t - 1$ but non-necessarily zero (e.g., $\sqrt{t}$, or $t/2$), which might be sufficient to establish non-trivial lower bounds. Let $\mathcal{T} = (\mathcal{I}, \mathcal{O}, \Delta)$ be a task for $n$ processes, and let $\mathcal{M}$ be a communication model with $\Xi$ its associated communication map. For defining a speedup task $\mathcal{T}' = (\mathcal{I}, \mathcal{O}', \Delta')$, we define the complex $\mathcal{O}'$,
On Extending Brandt’s Speedup Theorem

We show that, under certain conditions, the task $T'$ are a subset of all the pairs $(i, P_i)$ with $P_i = (x_i, S_i), x_i \in \text{val}(I)$, and

$$S_i = \{ (S_{i,e,E_i} : (i, E_i) \in \mathcal{M}) \land (e \subseteq [n]) \land (i \in e) \},$$

where, for every vertex $(i, E_i) \in \mathcal{M}$, and for every $e \subseteq [n]$ and $i \in e$, $S_{i,e,E_i} \subseteq 2^{\mathcal{S}_k(O)}$. In other words, $S_{i,e,E_i}$ is a collection of sets with elements in $\mathcal{S}_k(O)$. Note that $\mathcal{S}_k(O)$ is merely a set of vertices of $O$, each of the form $(i, y)$ for some $y \in \text{val}(O)$. There is a set $S_{i,e,E_i}$ for every set $E_i$ of channels incident to $i$ susceptible to be active in some communication, and for every potential channel $e \subseteq [n]$. In fact, each set $S_{i,e,E_i} \subseteq S_i$ is identified by a pair $(e, E_i)$. That is, formally, $S_i$ is an array indexed by pairs (channel, set of channels). Nevertheless, for the sake of simplifying the notations, we describe $S_i$ as a set. For $(i, (x_i, S_i))$ to be a vertex of $O'$, the sets $S_{i,e,E_i}$ in $S_i$ must satisfy the following property:

**P0:** For every vertex $(i, E_i) \in \mathcal{M}$ with $E_i = \{ e_1, \ldots, e_d \}$, and for every $(S_{i,e_1,E_i}, \ldots, S_{i,e_d,E_i}) \in S_{i,e_1,E_i} \times \ldots \times S_{i,e_d,E_i}$, we have $\bigcap_{j \in [d]} S_{i,e_j,E_i} \neq \emptyset$.

**P1:** For every $e = \{ i_1, \ldots, i_d \} \in E_i$, there exists $(S_{i_{1,e_1,E_{11}}, \ldots, S_{i_{d,e_d,E_{id}}}}) \in S_{i_{1,e_1,E_{11}}} \times \ldots \times S_{i_{d,e_d,E_{id}}}$ such that, for every $\{(i_1, y_1), \ldots, (i_d, y_d)\} \in S_{i_{1,e_1,E_{11}}} \times \ldots \times S_{i_{d,e_d,E_{id}}}$, there exists $\tau \in I$ for which

$$\{(i_1, x_{i_1}), \ldots, (i_d, x_{i_d})\} \subseteq \tau,$$

and $\{(i_1, y_1), \ldots, (i_d, y_d)\} \in \text{Cl}(\Delta(\tau))$.

Moreover, for every $k \in [d]$,

$$\forall E_k \in \text{Cl}(\text{St}(i_k, E_k)), S_{i_k,e,E_i,E_{ik}} = S_{i_k,e,E_i,E_{ik}}$$

**P2:** $|S_{i,(i_1), E_i}| = 1$ (i.e., there is a unique set in $S_{i,(i_1), E_i}$), and, for every $j \in I$, for every $e \in E_j$, if $i \in e$ and $j \notin \cup_{e \in E_j} f$, that is, if $p_i$ does not receive information from $p_j$ in any channel of $E_j$, then $S_{i,e,E_i} = S_{i,(i_1), E_i}$.

Finally, the input-output specification $\Delta'$ satisfies the following:

**P3:** For every two simplices $\sigma = \{(i, x_i) : i \in I\} \in I$, and $\tau = \{(i, (x_i', S_i)) : i \in I\} \in O'$, where $I \subseteq [n]$,

$$\tau \in \Delta'(\sigma) \iff \forall i \in I, x_i' = x_i.$$

### 5.1 From $t$ Rounds to $t - 1$ Rounds

We show that, under certain conditions, the task $T' = (I, O', \Delta')$ defined above is a weak speedup of $T$, that is it verifies the first condition of the definition [13]. The main condition is actually not local checkability, but a local independence property. This property is related to the protocol complex at time $t$, and is local to every process. Given a vertex $(i, E_i) \in \mathcal{M}$ and a channel $e \in E_i$, let us consider a simplex $\{(j, v_j) : j \in e\} \in \mathcal{P}^{(t)}$. The views $v_j$ at time $t$ of the processes $j \in e$ are mutually consistent. Let $j \in e$, and let us recall that $\text{St}(j, v_j)$ denotes the complex induced by the set of all simplices of $\mathcal{P}^{(t)}$ containing vertex $(j, v_j)$. For every $j \in e$, and every channel $f \in E_j \setminus \{e\}$, one can consider a simplex
We have for every closed simplex $\{ (k, v_k) : k \in f \} \in \text{St}(j, v_j)$. The $t$-independence property essentially states that the simplices $\sigma_{f,j}$, for $j \in e$ and $f \in E_j \setminus \{e\}$ are “independent” of each other, in the sense that, if, for every $j \in e$ and every $f \in E_j \setminus \{e\}$, the views of the processes $k \in f$ are consistent with the view of process $j \in e$, then the views of all these processes are mutually consistent all together. That is, the union of all the simplices $\sigma_{f,j}$ is a simplex of $\mathcal{P}(t)$.

**Definition 15.** Let $t \geq 1$ be an integer. $T$ satisfies the $t$-independence property w.r.t. $M$ if, for every closed simplex $\{(i, E_i) : i \in I\} \in M$, for every $i \in I$, for every $e \in E_i$, for every simplex $\{(j, v_j) : j \in e\} \in \mathcal{P}(t)$, and for every collection

$$\{\sigma_{f,j} = \{(k, v_k) : k \in f\} \in \text{St}(j, v_j) : (j \in e) \land (f \in E_j \setminus \{e\})\},$$

we have

$$\bigcup_{j \in e} \bigcup_{f \in E_j \setminus \{e\}} \sigma_{f,j} \in \mathcal{P}(t). \quad (1)$$

Note that the $t$-independence property depends solely on the model $M$, and on the input complex $I$ of the task. Indeed, $I$ and $M$ are the only parameters that govern the properties of the protocol complex at time $t$. An interesting special case of the $t$-independence property is when considering the “self-loop” $e = \{i\}$ (recall that we assumed that $\{i\} \in E_i$ for every $i \in E_i \in M$). For the self-loops, the $t$-independence property implies that, for every $(i, E_i) \in M$, for every vertex $(i, v_i) \in \mathcal{P}(t)$, and for every collection $\{\sigma_f = \{(j, v_j) : j \in f\} \in \text{St}(i, v_i) : f \in E_i \setminus \{i\}\}$, we have

$$\bigcup_{f \in E_i \setminus \{i\}} \sigma_f \in \mathcal{P}(t). \quad (2)$$

**Example** Let us consider LOCAL($G$) where $G$ is the graph on $n = 2k$ nodes $u_1, \ldots, u_k, v_1, \ldots, v_k$, with edges $\{u_1, v_1\}$ and $\{u_i, v_i\}$ for $i = 1, \ldots, k$. In other words, $G$ consists of two stars centered at $u_1$ and $v_1$, respectively, and sharing the edge $\{u_1, v_1\}$. In $G$, for $i \in [k]$, process $2i - 1$ occupies node $u_i$, and process $2i$ occupies node $v_i$. Let $T = (I, \mathcal{O}, \Delta)$ be a task where $I$ is the input complex corresponding to $3$-coloring (without IDs). Specifically, $\rho = \{(i, x_i) : i \in [n]\}$ is a facet of $I$ if, for every $i \in [n]$, $x_i \in \{1, 2, 3\}$, and, for every $i \in [k]$, $x_1 \neq x_2$, and $x_2 \neq x_{2i-1}$.

We illustrate 0-independence by demonstrating that, for $t = 0$, Eq. $(1)$ holds for $p_1$, and for the channel $e$ between $p_1$ and $p_2$ (the extension to other processes, and to other channels is straightforward). As a warm up, we start by showing that Eq. $(2)$ holds for $p_1$. Process 1 is occupying node $u_1$, and we have

$$E_1 = \{(1), \{1, 2\}, \{1, 4\}, \ldots, \{1, 2k\}\}.$$  

Let $f \in E_1 \setminus \{1\}$, say $f = \{1, 2i\}$ for some $i \in [k]$, and let us assume that $p_1$ is colored $x_1 \in \{1, 2, 3\}$. The set $\sigma_f = \{(1, x_1), (2i, x_{2i})\}$ is a simplex of $\mathcal{P}(0) = \mathcal{I}$ in $\text{St}(1, x_1)$ whenever $x_{2i} \in \{1, 2, 3\} \setminus \{x_1\}$. We have

$$\bigcup_{f \in E_1 \setminus \{1\}} \sigma_f = \{(1, x_1), (2, x_2), (4, x_4), \ldots, (2k, x_{2k})\},$$

and indeed this simplex is in $\mathcal{I}$ as $x_{2i} \neq x_1$ for every $i \in [k]$. Thus Eq. $(2)$ is satisfied by $p_1$. Let us now turn our attention to the channel $e = \{1, 2\} \in E_1$, and let us assume that $p_1$ is colored $x_1 \in \{1, 2, 3\}$ while $p_2$ is colored $x_2 \in \{1, 2, 3\} \setminus \{x_1\}$. Let $j \in e = \{1, 2\}$, and
On Extending Brandt’s Speedup Theorem

Let \( f \in E_j \setminus \{e\} \). For \( j = 1, f = \{1, 2i\} \) for some \( i \in [2, k] \), and the set \( \sigma_{f,1} = \{(1,x_1),(2i,x_{2i})\} \) is a simplex of \( \mathcal{P}^{(0)} = \mathcal{I} \) in \( \text{St}(1,x_1) \) whenever \( x_{2i} \in \{1, 2, 3\} \setminus \{x_1\} \). Similarly, for \( j = 2, f = \{2, 2i - 1\} \) for some \( i \in [2, k] \), and the set \( \sigma_{f,2} = \{(2,x_2),(2i-1,x_{2i-1})\} \) is a simplex of \( \mathcal{P}^{(0)} = \mathcal{I} \) in \( \text{St}(2,x_2) \) whenever \( x_{2i-1} \in \{1, 2, 3\} \setminus \{x_2\} \).

We have

\[
\bigcup_{j \in \{1,2\}} \bigcup_{f \in E_j \setminus \{(1,2)\}} \sigma_{f,j} = \{(1,x_1),(2,x_2),(3,x_3),(4,x_4),(5,x_5),\ldots,(2k,x_{2k})\}.
\]

This simplex is actually a facet of \( \mathcal{I} \), and thus is in \( \mathcal{P}^{(0)} = \mathcal{I} \). Thus Eq. (1) is satisfied by \( p_1 \) and the channel \( e = \{1,2\} \in E_1 \). The same argument can be used to show that Eq. (1) is satisfied for every process \( i \in [n] \), for any channel \( e \in E_i \), which demonstrates 0-independence.

This example can be extended to the graph \( G \) consisting of two full \((k-1)\)-ary trees of height \( h > 1 \) (each node has \( k-1 \) children, excepted the leaves, which are all at distance \( h \) from the root), connected by an edge \( e = \{i,j\} \) connecting process \( i \) to process \( j \). For instance, for \( t \leq h - 1 \), and for a vertex \((i,v_i)\) of \( \mathcal{P}^{(t)} \), the view \( v_i \) is a tree rooted at \( p_i \) spanning all nodes of \( G \) at distance at most \( t \) from \( p_i \), whose nodes are properly 3-colored. A set \( \{((i,v_i),(j,v_j))\} \) is a simplex of \( \mathcal{P}^{(t)} \) whenever the two views \( v_i \) and \( v_j \) are compatible.

That is, all processes in \( v_i \cap v_j \) are colored the same in the two views \( v_i \) and \( v_j \), and the colors of processes \( p_i \) and \( p_j \) are different. In this context, the \( t \)-independence property essentially says that whatever color is given to each node not in \( v_i \cup v_j \) but adjacent to a leaf of \( v_i \) or \( v_j \), if this color is different from the color of the leaf it is attached to, the resulting set

\[
\{(i,v_i),(j,v_j)\} \cup \{(k,v_k) : k \in (N_G(i) \cup N_G(j)) \setminus \{i,j\}\},
\]

where, for every \( k \in (N_G(i) \cup N_G(j)) \setminus \{i,j\} \), \( v_k \) is the view at distance \( t \) of a neighbor of \( p_i \) or \( p_j \) that is compatible with \( v_i \) and \( v_j \), with the nodes not in \( v_i \cup v_j \) colored arbitrarily as above, is indeed a simplex of \( \mathcal{P}^{(t)} \).

The following lemma does not require local checkability, but solely the local independence property.

**Lemma 16.** Let \( T = (\mathcal{I}, \mathcal{O}, \Delta) \) be a task for \( n \) processes, and let \( M \) be a model on \( n \) vertices. Let \( t \geq 1 \) be an integer, and assume that \( T \) satisfies the \((t-1)\)-independence property w.r.t. \( M \). If \( T \) is solvable in \( t \) rounds in \( M \), then the task \( T' = (\mathcal{I'}, \mathcal{O'}, \Delta') \) is solvable in \( t - 1 \) rounds in \( M \), where \( \mathcal{O'} \) is the complex defined by properties P0-2, and \( \Delta' \) is the input-output relation defined by P3.

**Proof.** Let \( \delta : \mathcal{P}^{(t)} \to \mathcal{O} \) solving \( T \) in \( t \) rounds in \( M \). We define \( \alpha : \mathcal{P}^{(t-1)} \to \mathcal{O'} \), and then show that \( \alpha \) is solving \( T' \) in \( M \). Intuitively the algorithm corresponding to \( \alpha \) consists in simulating every possible solution for \( T \) when the values in \( \mathcal{P}^{(t-1)} \) of some subset of processes are fixed, and when the communications occurring at time \( t \) are fixed. For a closed simplex \( \varphi \in M \), and a chromatic complex \( K \), we define

\[
\Xi(K, \varphi) = \text{Cl} \left( \bigcup_{\sigma \in K, \text{name}(\sigma) = \text{name}(\varphi)} \Xi(\sigma, \varphi) \right).
\]

Formally, note that, for any two closed simplices \( \varphi \) and \( \psi \) in \( \text{St}(i, E_i) \), \( \text{Sk}_i(\Xi(\text{St}(\sigma), \varphi)) = \text{Sk}_i(\Xi(\text{St}(\sigma), \psi)) \) for every simplex \( \sigma \in \mathcal{P}^{(t-1)} \). Therefore, we can abuse notation by denoting \( \text{Sk}_i(\Xi(\text{St}(\sigma), \varphi)) \) as \( \text{Sk}_i(\Xi(\text{St}(\sigma), E_i)) \). For any vertex \((i,v_i) \in \mathcal{P}^{(t-1)} \), we let

\[
\alpha(i,v_i) = (i, P_i) \quad \text{with} \quad P_i = (x_i, S_i),
\]
Therefore, which is indeed of dimension $\dim(S) = |e| - 1$. For every $\sigma \in \text{Sk}_e(\Xi(\text{St}(\sigma), E_i))$, we have $\text{Sk}_e(\Xi(\text{St}(\sigma), E_i)) \subseteq \text{Sk}_i(\Xi(\text{St}(\sigma), E_i))$. Therefore, property P0 holds, that is, $\delta(\text{Sk}_i(\Xi(\text{St}(\sigma), E_i))) \subseteq \bigcap_{e \notin E_i} S_{i,e,E_i}$. Now, $\delta(\text{Sk}_i(\Xi(\text{St}(\bigcup_{e \in E_i} \sigma_e), E_i)))$ cannot be empty, simply because $\bigcup_{e \in E_i} \sigma_e \in \mathcal{P}^{t-1}$. Therefore, property P0 holds by construction.

To prove that $\alpha$ solves $T'$, it is sufficient to consider an arbitrary facet $\rho = \{(i, v_i) : i \in [n] \} \in \mathcal{P}^{t-1}$, and its image $\alpha(\rho) = \{(i, x_i, S_i) : i \in [n] \}$, and we show that $\alpha(\rho)$ is a facet of $\mathcal{O}'$ that agrees with $\Delta'$. It is sufficient to show that both properties P1 and P2 hold as, by definition of $\alpha$, P3 holds by construction.

First we prove that P1 holds. For every closed simplex $\varphi = \{(i, E_i) : i \in I \} \in \mathcal{M}$, and for every $e = \{i_1, \ldots, i_d\} \in E_i$ for some $i \in I$, we consider the face $\sigma = \{(i_k, v_k) : k \in [d] \}$ of $\rho$, which is indeed of dimension $|e| - 1$. Note that, since $(i_k, v_k) \in \sigma$ for every $k \in [d]$, we have $\sigma \in \bigcap_{k=1}^{d} \text{Sk}_e(\text{St}(i_k, v_k))$. For every $k \in [d]$, let us consider the set

$$S_{i_k,e,E_{i_k}}^\sigma = \delta(\text{Sk}_i(\Xi(\text{St}(\sigma), E_{i_k}))) \in S_{i,k,e,E_{i_k}}.$$
Note that $S^\sigma_{\ell t,c,E_i}$ is by definition independent of $E_i$ for $\ell \in [d] \setminus \{k\}$ therefore it is enough to show that $S^\sigma_{\ell t,c,E_i}$ satisfies the first part of P1 and let $((i_1, y_1), \ldots, (i_d, y_d)) \in S^\sigma_{\ell t,c,E_i}$, $\ell$ and $d$. Let $k \in [d]$. By definition of $\alpha$, there exists $\alpha_k \in \text{St}(\sigma)$ with name$(\alpha_k) = (\cup f \in E_i) \setminus c$ such that $(i_k, y_k) \in \delta(Sk_i(\Xi(\sigma_k \cup \sigma, E_i)))$. In fact, if we fix the communication pattern $E_i\ell$ occurring at time $t$, and if we fix the values in $P^{(t-1)}$ of all the processes in the channels of $E_i\ell$, there exists exactly one vertex $(i_k, w_k) \in P^{(t)}$ that is consistent with the fixed communication pattern, and with the fixed values in $P^{(t-1)}$. Therefore the fixed values yield exactly one output for process $i_k$. In other words, for every $k \in [d]$, 

$$\delta(Sk_i(\Xi(\sigma_k \cup \sigma, E_i))) = \{(i_k, y_k)\}.$$ 

Using the $(t-1)$-independence property, it holds that $(\cup_{k=1}^d \sigma_k) \cup \sigma \in P^{(t-1)}$. It follows that, for every $k \in [d]$, we have 

$$\emptyset \neq \delta(Sk_i(\Xi((\cup_{k=1}^d \sigma_k) \cup \sigma, E_i))) \subseteq \delta(Sk_i(\Xi(\sigma_k \cup \sigma, E_i))) = \{(i_k, y_k)\}$$ 

As a consequence, 

$$\delta(Sk_{\{i_1, \ldots, i_d\}}(\Xi((\cup_{k=1}^d \sigma_k) \cup \sigma, \varphi))) = \{(i_k, y_k) : k \in [d]\}.$$ 

Recall that, by definition of $\alpha$, for any $j \in [d]$, $\alpha(i_j, v_{i_j}) = (i_j, (x_{i_j}, S_{i_j}))$, where $x_{i_j} \in \text{val}(I)$ is the input of process $i_j$ in $v_{i_j}$. Let us consider a facet $\phi$ of $I$ such that $(\cup_{k=1}^d \sigma_k) \cup \sigma \in \Delta(\phi)$. Note that, in particular, $\{(i_j, x_{i_j}) : j \in [d]\} \subseteq \phi$. Moreover, since $\delta$ solves $T$, we also have $\{(i_k, y_k) : k \in [d]\} \in \text{Cl}(\Delta(\phi))$. It follows that Property P1 holds.

Second, we show that P2 holds. For every closed simplex $\varphi = \{(i, E_i) : i \in I\} \in M$, for every $i \in I$ and for every $e \in E_j$, if $i \in e$ and $j \notin \cup f \in E_j f$, we show that 

$$S_{i,e,E_i} = S_{i,(i),E_i} \overset{=} {\triangleq} \{\delta(Sk(\Xi(\text{St}(i, v_i), E_i)))\}.$$ 

Note that $|S_{i,(i),E_i}| = 1$. Let $S \in S_{i,e,E_i}$. We show that $S = \delta(Sk(\Xi(\text{St}(i, v_i), E_i)))$. By definition of $S_{i,e,E_i}$, there exists a simplex $\sigma_0 \in Sk_c(\text{St}(i, v_i))$ such that $S = \delta(Sk(\Xi(\text{St}(\sigma_0), E_i)))$. Using the $(t-1)$-independence property on $e$, it follows that, for every $\sigma \in Sk_c(\text{St}(i, v_i))$, and for every collection $\{\sigma_f (f \in Sk(\text{St}(i, v_i)))) : f \in E_i \setminus e\}$ of simplices, 

$$\left( \bigcup_{f \in E_i \setminus e} \sigma_f \right) \cup \sigma \in P^{(t-1)}.$$ 

Now, $j \in e$ but, for every $f \in E_i$, $j \notin f$. It follows that $e \notin E_i$, and thus $E_i \setminus e = E_i$. Therefore $\cup f \in E_i \setminus e \sigma_f$ defines the values of all the processes $k$ that are in at least one channel of $E_i$. This entirely characterizes the output of the process $i$ by $\delta$. Therefore, we have established that $\delta(Sk(\Xi(\text{St}(\sigma_0), E_i)))$ is actually independent of the simplex $\sigma_0 \in Sk_c(\text{St}(i, v_i))$. Formally, 

$$\delta(Sk(\Xi(\text{St}(\sigma_0), E_i))) = \delta(Sk(\Xi(\text{St}(i, v_i), E_i))),$$ 

which implies that P2 holds, and concludes the proof. \hfill \blacktriangleleft

### 5.2 From $t-1$ Rounds to $t$ Rounds

It follows from Lemma [10] that the weak version of Brandt’s speedup Theorem can be extended from LOCAL to asynchronous computing models with crashes. The strong version
of the speedup theorem does not only require the local independence property, but also the edge-checkability property, a stronger variant of local decidability (cf. Def. 17).

Granted with the framework of Section 3 we define a general notion of local decidability (also referred to as local checkability). We generalize the specific notion defined for LOCAL (see 19), and the specific notion defined for WAIT-FREE (see 21). Given a simplex \( (i, v_i) : i \in I \) of a state complex \( K \), and given \( J \subseteq I \), let \( \pi_J(\sigma) = \{(i, v_i) : i \in J\} \). Note that \( \pi_J(\sigma) \) is different from \( S \), since it is a simplex, while the former is a complex.

**Definition 17.** A task \( (I, O, \Delta) \) is locally checkable for the communication model \( M \) if, for every \( \sigma \in I \) with \( \text{name}(\sigma) = I \), for every set \( \tau = \{(i, y_i) : i \in I \land y_i \in \text{val}(O)\} \), and for every closed simplex \( \varphi = \{(i, E_i) : i \in I\} \) of \( M \), the following holds:

\[
\tau \in \Delta(\sigma) \iff \forall i \in I, \quad \pi_{J_i}(\tau) \in \Delta(\pi_{J_i}(\sigma)), \quad \text{where } J_i = \cup_{e \in E_i} e.
\]

The class of locally checkable tasks for the model \( M \) is denoted by \( LD(M) \). A problem \( \Pi \) is locally checkable if, for every \( (T, M) \in \Pi \), the task \( T \in \text{LD}(M) \).

In other words, a task \( (I, O, \Delta) \) is in \( \text{LD}(M) \) if the correctness of a potential solution \( \tau = \{(i, y_i) : i \in I\} \in O \) for an input \( \sigma = \{(i, x_i) : i \in I\} \in I \) can be checked in a single round of communication under \( M \). Indeed, assuming every process \( i \in I \) is given a pair \((x_i, y_i)\) of input-output values, any round of communication performed according to some communication pattern \( \varphi \in M \) allows every process \( i \in I \) to acquire a set \( \{\pi_{J_i}(\tau) : j \in J_i\} \) where \( x_j \) and \( y_j \) are the input and output values of process \( j \in J_i \) in \( \sigma \) and \( \tau \), respectively. Every process \( i \in I \) can thus check whether \( \pi_{J_i}(\tau) \in \Delta(\pi_{J_i}(\sigma)) \) or not. Locally decidability states that the output \( \tau \) is correct for the input \( \sigma \), where \( \text{name}(\sigma) = \text{name}(\tau) = I \subseteq [n] \), if and only if all the individual tests performed by the processes in \( I \) are passed. In LOCAL, many standard graph problems, e.g., vertex coloring and maximal independent sets (MIS), are locally checkable. Similarly, \((d + 1)\)-coloring is in \( \text{LD}(\text{WF-LOCAL}(G)) \) for every \( G \in G_d \). This is because the model guarantees that, for every edge \((i, j)\), at least one of the two processes \( i \) and \( j \) receives the color of the other process.

On the other hand, as opposed to the case of \( \text{LOCAL}(G) \), MIS is not in \( \text{LD}(\text{WF-LOCAL}(G)) \). Indeed, a node that is not in the set may not receive information from all its neighbors, and thus it cannot systematically check whether it has at least one neighbor in the set. The generalized maximal independent set (GMIS) task \( \text{GMIS} \) is however in \( \text{LD}(\text{H-LOCAL}(H)) \) for every hypergraph \( H \) in GMIS, each hyperedge \( e \) is associated with a threshold \( t_e \in \{1, |e| - 1\} \), and a set \( S \) of vertices is a solution to GMIS if, for every \( e \in E(H) \), \( |e \cap S| \leq t_e \), and \( S \) is maximal for this property.

**Definition 18.** A task \( (I, O, \Delta) \) is edge-checkable in a model \( M \) if, for every \( \sigma \in I \) with \( \text{name}(\sigma) = I \), for every set \( \tau = \{(i, y_i) : i \in I \land y_i \in \text{val}(O)\} \), and for every closed simplex \( \varphi = \{(i, E_i) : i \in I\} \) of \( M \), the following holds:

\[
\tau \in \Delta(\sigma) \iff \forall i \in I, \; \forall e \in E_i, \; \pi_{e}(\tau) \in \Delta(\pi_{e}(\sigma)).
\]

Note that MIS (in its standard form) is locally checkable in LOCAL. However, MIS is not edge-checkable in LOCAL because, by considering each of its neighbors independently, a process that is not in the MIS cannot determine whether it has at least one neighboring process that is in the MIS. Nevertheless we describe further a systematic way to transform a locally checkable task into an edge-checkable task, which applies to LOCAL, as well as to other models. As in 8, our speedup theorem assumes an underlying mechanism enabling the processes in a same channel \( e \) to break symmetry, whether it be thanks to a local or
global identification mechanism, or thanks to an implicit or explicit ordering of the processes in the same channel.

**Theorem 19.** Let $T = (\mathcal{I}, \mathcal{O}, \Delta)$ be a task for $n$ processes, and let $M$ be a model on $n$ vertices that supports symmetry breaking in its channels. Let $t \geq 1$ be an integer. Let us assume that $T$ satisfies the $(t-1)$-independence property w.r.t. $M$, and that $T$ is edge-checkable in $M$. $T$ is solvable in $t$ rounds in $M$ if and only if $T' = (\mathcal{I}, \mathcal{O}', \Delta')$ is solvable in $t-1$ rounds in $M$, where $\mathcal{O}'$ is the complex defined by properties P0-2, and $\Delta'$ is the map defined by Property P3.

**Proof.** Thanks to Lemma 16, it is sufficient to show the existence of a simplicial map $\beta : \Xi(\mathcal{O}') \to \mathcal{O}$ such that, for every closed simplex $\sigma \in \mathcal{I}$, $\beta(\Xi(\Delta'(\sigma))) \subseteq \Delta(\sigma)$. Let $\tau = \{(i, (x_i, S_i)) : i \in I\} \in \Delta'(\sigma)$ for some closed simplex $\sigma = \{(i, x_i) : i \in I\} \in \mathcal{I}$, note that $\tau$ is a face of a facet of $\mathcal{O}'$, which, by definition, satisfy P1 and P2. Since $T$ is edge-checkable, it is sufficient to prove that, for every closed simplex $\varphi = \{(i, E_i) : i \in I\} \in M$, for every $i \in I$, process $i$ can output a solution $(i, y_i) \in Sk_i(O)$ such that, for every $e \in E_i$, $\{(j, y_j) : j \in e\} \in \Delta(\pi_e(\sigma))$. After one round of communication according to $\varphi$, every process $i \in I$ receives messages from every process $j \in e$ for every $e \in E_i$. Therefore every process $i$ has access to the set $\{(x_k, S_k) : k \in J_i\}$ where $J_i = \cup_{e \in E_i} e$. More specifically, for every $e \in E_i$, process $i$ has access to $\{(x_k, S_k) : k \in e\}$. If $j \in e$ does not receive from $i$, property P2 guarantees that $|S_{j,e,E_i}| = 1$. For the other processes $j \in e$, the symmetry-breaking mechanism and property P1 allows these processes to choose sets $S_{j,e,E_i} \subseteq S_{j,e,E_j}$ such that, for every choice of $\{y_j : j \in e\}$ where $(j, y_j) \in S_{j,e,E_j}$ for every $j \in e$, there exists $\tau \in \mathcal{I}$ such that

$$\{(j, y_j) : j \in \tau\} \subseteq \tau,$$

and $\{(i, y_i) : j \in e\} \in \Delta(\tau)$. Since $T$ is edge-checkable, $\{(j, y_j) : j \in e\} \in \Delta(\tau)$ implies that

$$\{(j, y_j) : j \in \tau\} \in \Delta(\pi_e(\tau)) = \Delta(\pi_e(\sigma)).$$

By repeating this operation for every channel $e \in E_i$, process $i$ can output any value $y_i$, such that

$$(i, y_i) \in \bigcap_{e \in E_i} S_{i,e,E_i}.$$ Such a value $y_i$ does exist thanks to property P0. The correctness of this algorithm is straightforward since the task is edge-checkable, and the set $S_{i,e,E_i}$ are precisely chosen to satisfy P1.

**5.3 Applications**

As mentioned before, several models satisfy the conditions in the statement of Theorem 19 beyond LOCAL. This is for instance the case of dynamic graph models, $\text{DYN}(\mathcal{F})$, under some conditions on $\mathcal{F}$. This is also the case of hypergraph models, $\text{H-LOCAL}(H)$, under some conditions on $H$. This is even the case for asynchronous crash-prone models such as $\text{WAIT-FREE}$ with 2 processes, this has already been detailed in Section 2.5.1. Therefore we focus here on giving a complete proof of the application of Theorem 19 to $\text{H-LOCAL}$.

First, we show that, in $\text{H-LOCAL}$, edge-checkability is actually essentially equivalent, up to $\pm 1$ rounds, to local decidability (see proof in Appendix C).
Lemma 20. Let $H$ be a hypergraph on $n$ nodes. For every task $T = (I, O, \Delta)$ on $n$ processes, if $T$ is locally checkable in $H$-LOCAL$(H)$, then there exists a task $T' = (I, O', \Delta')$ that is edge-checkable in $H$-LOCAL$(H)$, such that (1) any solution for $T$ can be transformed into a solution for $T'$ via a single round of communication in H-LOCAL$(H)$, and (2) any solution for $T'$ can be transformed into a solution for $T$ in zero rounds.

The following is a direct consequence of Theorem 19.

Corollary 21. Let $H$ be a hypergraph on $n$ nodes, let $T = (I, O, \Delta)$ be a task for $n$ processes, and let $t \geq 1$. Let us assume that $T$ satisfies the $r$-independence property w.r.t. $H$-LOCAL$(H)$ for every $r \in \{0, \ldots, t-1\}$, and that $T$ is edge-checkable in $H$-LOCAL$(H)$. Let us assume the existence of a symmetry breaking mechanism among the processes in each hyperedge of $H$. $T$ is solvable in $t$ rounds in $H$-LOCAL$(H)$ if and only if $T(t) = (I, O'(t), \Delta'(t))$ is solvable in zero rounds in $H$-LOCAL$(H)$, where $T(t)$ is the task defined by iterating $t$ times the construction defined by properties P0-3.

Proof. Thanks to Theorem 19, it is sufficient to prove that the task $T' = (I, O', \Delta')$ obtained by applying the construction defined by properties P0-3 still satisfies the precondition in the statement of the theorem. First, $T'$ is edge-checkable. Indeed, the conditions specified by P0-3 are locally checkable, as the communication pattern is unique in $H$-LOCAL$(H)$, and Property P2 is trivially satisfied in $H$-LOCAL$(H)$ as the model in undirected (all processes in a hyperedge $e$ play the same role, as far as the channel $e$ is concerned). $T'$ satisfies the $(t-2)$-independence property w.r.t. $H$-LOCAL$(H)$, simply because $T$ itself satisfies the $(t-2)$-independence property, and $T$ and $T'$ have the same input complexes $I$.

For instance, let us consider $H$-LOCAL$(H)$ where $H$ is a linear hypergraph [31], that is, for every two distinct hyperedges $e$ and $f$, $|e \cap f| \leq 1$. Let us also assume that $H$ is locally isomorphic to a (linear) hypertree. Formally, $H$ has no small cycle, that is, its girth is greater than $2t + 1$. Linear hypertrees can be viewed as graphs where hyperedges are displayed as cliques (cf. Fig. 6 for an example). Note that any lower-bound for $H$-LOCAL$(H)$ holds for LOCAL$(G)$ where $G$ is obtained from $H$ by replacing the hyperedges of $H$ by cliques in $G$. To show that the $r$-independence property holds for any $r \geq 0$ in such a hypergraph, let us fix an hyperedge $e \in E(H)$, and, for $i \in e$ and $f \in E_i \setminus e$, let $N_r(i, f)$ denote the sub-hypertree induced by all the nodes of $H$ at distance at most $r$ from node $i$ via the edge $f$. The sub-hypertrees $N_r(i, f)$ for $i \in e$ and $f \in E_i \setminus e$ are vertex-disjoint, and thus the $r$-independence property is satisfied in absence of labeling, or with an edge-checkable labeling. The class of linear hypergraphs with large girth contains the graphs that are locally isomorphic to a regular tree, which is precisely the class of graphs on which Brandt’s speedup theorem [8] has been originally proven. However, linear hypergraphs form a much larger class, which includes graphs with short cycles, like the one depicted on Fig. 6.

6 Conclusion

In this paper, we have extended Brandt’s speedup theorem from LOCAL to round-based based communication models supporting full-information protocols, including many standard synchronous communication models in networks, and even asynchronous models such as WF-LOCAL, and WAIT-FREE for 2 processes. Extending the speedup theorem to WAIT-FREE for more than 2 processes remains open. Our approach consisted in decomposing every subcomplex $\delta(\Xi(St(i, v_i)))$ of the output complex, where $(i, v_i)$ is a vertex of $\mathcal{P}^{t-1}$, into collections of sets $S_{i, e, E_i}$, one for each potential channel $e$, and for each possible local
communication pattern \( E \) of the underlying communication model \( M \). This approach is well suited to models like LOCAL, H-LOCAL, or even DYN and WAIT-FREE for two processes, but it does not match the characteristics of WAIT-FREE for larger systems, essentially because WAIT-FREE does not satisfy the local independence property whenever \( n > 2 \). Nevertheless, we believe that there is another way to decompose the complexes \( \delta(\Xi(St(i,v_i))) \), for all \( (i,v_i) \in P^{(-1)} \), that would provide a speedup theorem for model not satisfying local independence (e.g., WAIT-FREE), but this decomposition still remains to be found.

References

1. Manuel Alcantara, Armando Castañeda, David Flores-Peñaloza, and Sergio Rajsbaum. The topology of look-compute-move robot wait-free algorithms with hard termination. *Distributed Comput.*, 32(3):235–255, 2019.
2. Hagit Attiya, Armando Castañeda, Maurice Herlihy, and Ami Paz. Bounds on the step and namespace complexity of renaming. *SIAM J. Comput.*, 48(1):1–32, 2019.
3. Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Series on Parallel and Distributed Computing. Wiley, 2004.
4. Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The distributed complexity of locally checkable problems on paths is decidable. In *38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 262–271, 2019.
5. Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
6. Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 481–497, 2019.
7. Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems? In *39th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 299–308, 2020.
8. Sebastian Brandt. An automatic speedup theorem for distributed problems. In *38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 379–388, 2019.
9. Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. A topological perspective on distributed network algorithms. *Theor. Comput. Sci.*, 849:121–137, 2021.
10. Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: the lower bound. *Distributed Comput.*, 22(5-6):287–301, 2010.
11. Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: The upper bound. *J. ACM*, 59(1):3:1–3:49, 2012.
12. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *10th International Conference on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW)*, volume 6811 of *Lecture Notes in Computer Science*, pages 346–359. Springer, 2011.
13. Yi-Jun Chang. The complexity landscape of distributed locally checkable problems on trees. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
14. Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM J. Comput.*, 48(1):122–143, 2019.
15. Lisbeth Fajstrup, Eric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raussen. *Directed Algebraic Topology and Concurrency*. Springer, 2016.
16 Afonso Ferreira. Building a reference combinatorial model for manets. *IEEE Netw.*, 18(5):24–29, 2004.
17 Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–191. IEEE Computer Society, 2017.
18 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
19 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35:1–35:26, 2013.
20 Pierre Fraigniaud and Ami Paz. The topology of local computing in networks. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPIcs*, pages 128:1–128:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
21 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. Locality and checkability in wait-free computing. *Distributed Comput.*, 26(4):223–242, 2013.
22 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. A lower bound on the number of opinions needed for fault-tolerant decentralized run-time monitoring. *J. Appl. Comput. Topol.*, 4(1):141–179, 2020.
23 Emmanuel Godard and Eloi Perdereau. k-set agreement in communication networks with omission faults. In *20th International Conference on Principles of Distributed Systems (OPODIS)*, volume 70 of *LIPIcs*, pages 8:1–8:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
24 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory Comput.*, 12(1):1–33, 2016.
25 Éric Goubault, Samuel Mimram, and Christine Tasson. Geometric and combinatorial views on asynchronous computability. *Distributed Comput.*, 31(4):289–316, 2018.
26 Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey theory*, volume 20. John Wiley and Sons, 1990.
27 Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.
28 Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
29 Juho Hirvonen and Jukka Suomela. Distributed algorithms. Aalto University, Finland, 2020.
30 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010.
31 Fabian Kuhn and Chaodong Zheng. Efficient distributed computation of MIS and generalized MIS in linear hypergraphs. *CoRR*, abs/1805.03357, 2018.
32 Shay Kutten, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. Distributed symmetry breaking in hypergraphs. In *28th International Symposium on Distributed Computing (DISC)*, volume 8784 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 2014.
33 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.
34 Hammurabi Mendes, Christine Tasson, and Maurice Herlihy. Distributed computability in byzantine asynchronous systems. In *46th ACM Symposium on Theory of Computing (STOC)*, pages 704–713, 2014.
35 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995.
36 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Discrete Mathematics and Applications. SIAM, 2000.
37 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *52nd ACM Symposium on Theory of Computing (STOC)*, pages 350–363, 2020.
38 Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.
On Extending Brandt’s Speedup Theorem

We say that such maps are chromatic.

We consider a system of simplicial complexes where the closure of a set $A$.

Elements of Topology

A simplicial complex is defined by a vertex set $V$, and a collection $K$ of non-empty subsets of $V$, closed by inclusion. That is, if $σ ∈ K$, then every non-empty set $σ′ ⊆ σ$ belongs to $K$. Every set in $K$ is called a simplex. The set $V$ is often clear from the context, in which case one merely refers to a complex by the collection $K$, and to the vertex set of $K$ as $V(K)$.

The dimension of a simplex $σ$ is $|σ| − 1$, hence a vertex is a simplex of dimension zero. A face of a simplex $σ$ is any simplex $σ′ ⊆ σ$. A facet is a simplex that is maximal, i.e., not included in any other simplices. Note that a complex can be described by the list of its facets. A complex is pure if all its facets have the same dimension. The dimension of a pure complex is the dimension of any of its facets.

A sub-complex of a complex $K$ is a subset of $K$ that is a complex. The star of a simplex $σ$ in a complex $K$, denoted by $St(σ)$, is the set of simplices of $K$ having $σ$ as a face. The star of $σ$ naturally induces a sub-complex of $K$, composed of all simplices of $K$ included in at least one simplex of $St(σ)$. This complex is merely the closure, $Cl(St(σ))$, of the star of $σ$ in $K$, where the closure of a set $S$ of simplices of $K$, denoted by $Cl(S)$, is the smallest simplicial subcomplex of $K$ that contains each simplex in $S$. In fact, in this paper, we will abuse notation, and will refer to $St(σ)$ as a complex, that is, $St(σ)$ actually refers to $Cl(St(σ))$.

All complexes considered in this paper are chromatic. That is, each of their vertices has the form $(i, x)$, where $i ∈ [n] = \{1, \ldots, n\}$ is a process index (the “color” of the vertex), and $x$ is a value that depends on the context, and no simplices can contain two vertices with the same color. To avoid confusion when considering problems such as graph coloring, we refer to the index $i ∈ [n]$ of a process as its name (and not its color), and the non-empty index set $I ⊆ [n]$ of a simplex $σ = \{(i, x_i) : i ∈ I\}$ is denoted by name($σ$). Given a non-empty set $I ⊆ [n]$, $Sk_I(K)$ denotes the skeleton subcomplex of $K$ composed of all simplices $σ ∈ K$ with $\text{name}(σ) ⊆ I$.

A particular class of mappings between simplicial complexes plays a crucial role in the topological framework applied to distributed computing: those preserving simplices. Specifically, given two complexes $K$ and $K′$, a map $f : V(K) → V(K′)$ is simplicial if, for every $σ ∈ K$, $f(σ) ∈ K′$, where $f(σ) = \{f(v) : v ∈ σ\}$. All maps considered in this paper apply to chromatic complexes, and are name-preserving, i.e., $f(i, x) = (i, y)$ for every $(i, x) ∈ V(K)$. We say that such maps are chromatic.

Given a finite set $X$ of values, the $(n − 1)$-dimensional chromatic pseudo-sphere induced by $X$ is the complex $S_n(X)$ whose vertices are all pairs $(i, x)$ with $i ∈ [n]$ and $x ∈ X$, and
every non-empty set \( \{(i, x_i) : i \in I\} \) of vertices, where \( I \subseteq [n] \), forms a simplex. In short,  
\[
S_n(X) = \left\{ \{(i, x_i) : i \in I\} : (\emptyset \neq I \subseteq [n]) \wedge (\forall i \in I, x_i \in X) \right\}.
\]

### A.2 Distributed States

This section recalls how all possible states of a distributed system at a given time can be captured by a single combinatorial object, namely the state complex. Specific instantiations of state complexes are the input complexes, the output complexes, and the protocol complexes, described further in the text.

At any point in time, all possible global states of a distributed system can be represented as an \((n - 1)\)-dimensional complex. The vertices of this complex are of the form \((i, s)\) where \(i \in [n]\) is the name of a process, \(s \in X\) is a state of process \(i\), and \(X\) is the set of all possible states of a process. A non-empty set \(\{(i, s_i) : i \in I\}\) of such vertices, \(I \subseteq [n]\), forms a simplex if the states \(s_i, i \in I\), are mutually compatible. Mutual compatibility is a notion which depends on the context (e.g., input complex or output complex), and on the communication model (e.g., protocol complexes), but it should soon appear clear further in the paper. In general, a state complex of an \(n\)-process system with local states in \(X\) is a pure \((n - 1)\)-dimensional sub-complex of the chromatic pseudo-sphere \(S_n(X)\).

The fact that a vertex \((i, s)\) belongs to two different simplices \(\sigma\) and \(\sigma'\) with name(\(\sigma\)) = name(\(\sigma'\)) means that process \(i\) in local state \(s\) cannot distinguish the global state \(\sigma\) from the global state \(\sigma'\). More specifically, even if process \(i\) is aware that the system is in global state \(\sigma\) or \(\sigma'\), it remains uncertain about the local state of every other process \(j\) such that \((j, s') \in \sigma\) and \((j, s'') \in \sigma'\) with \(s' \neq s''\).

**Example.** A system maintaining a clock \(c_i \in \mathbb{F}_q\) at every process \(i \in [n]\), with bounded drift \(d\) between processes, i.e., \((c_i - c_j) \mod q \leq d\) for every \(i, j \in [n]\), has a state complex \(K \subseteq S_n(\mathbb{F}_q)\) where, for every non-empty \(I \subseteq [n]\): \(\{(i, c_i) : i \in I\} \in K \iff \forall i, j \in I, (c_i - c_j) \mod q \leq d\).

### A.3 Input and Output Complexes, and Distributed Tasks

This section recalls the important notion of task, which formalizes the typical “functions” to be computed in the distributed setting. A task is defined by three objects: the input complex, the output complex, and an input-output specification.

**Input Complex.** Let \(I\) be a finite set, whose elements are called input values. An input complex \(I\) with values in \(I\) is a pure \((n - 1)\)-dimensional sub-complex of the \((n - 1)\)-dimensional chromatic pseudo-sphere \(S_n(\mathbb{F})\). For instance, for binary consensus, \(I = \{0, 1\}\), and all sets \(\{(i, v_i) : i \in I\}\) with \(I \subseteq [n]\) and \(v_i \in \{0, 1\}\) for every \(i \in I\), are simplices of \(I\), i.e., \(I = S_n(\{0, 1\})\). Note that it is often the case that \(I = S_n(\mathbb{F})\), as in consensus. However, other problems assumes \(I \subset S_n(\mathbb{F})\). A typical example is \((m, k)\)-renaming, for \(m \geq k \geq n\), in which the \(n\) processes are given as input \(n\) distinct integers in the set \([m]\), and must output \(k\) distinct integers in \([k]\). In this case a non-empty set \(\{(i, v_i) : i \in I\}\) with \(I \subseteq [n]\) is a simplex of \(I\) if \(v_i \in [m]\) for every \(i \in I\), and \(v_i \neq v_j\) for every distinct \(i, j \in I\).

**Output Complex.** An output complex \(O\) is a pure \((n - 1)\)-dimensional sub-complex of the \((n - 1)\)-dimensional chromatic pseudo-sphere \(S_n(O)\) induced by a finite set \(O\), whose elements are called output values. For instance, in the case of binary consensus, \(O = \{0, 1\}\), and a non-empty set \(\{(i, v_i) : i \in I\}\) with \(I \subseteq [n]\), is a simplex of \(O\) if \(v_i \in \mathbb{O}\) for every \(i \in I\), and
On Extending Brandt’s Speedup Theorem

$v_i = v_j$ for every two $i, j \in I$. Note that the output complex has generally more structure than the input complex.

**Input-Output Relation.** The input-output relation specifies, for every input state, the collection of output states that are legal w.r.t. this input. Specifically, the input-output specification is a function $\Delta : I \rightarrow 2^O$ that is returning, for every input simplex $\sigma \in I$, a non-empty collection $\tau_1, \ldots, \tau_k$ of output simplices, $k \geq 1$, such that $\text{name}(\tau_i) = \text{name}(\sigma)$ for every $i = 1, \ldots, k$. As $\Delta$ is name-preserving, it is called chromatic. For instance, in the case of binary consensus, $\Delta$ maps every input simplex $\sigma = \{(i, v_i) : i \in I\} \in I$ to the output simplices $\tau_0 = \{(i, 0) : i \in I\}$ and $\tau_1 = \{(i, 1) : i \in I\}$ whenever there are two processes $i, j \in I$ with $v_i \neq v_j$ in $\sigma$. Instead, for every $x \in \{0, 1\}$, $\Delta$ maps the simplex $\sigma = \{(i, x) : i \in I\} \in I$ to the simplex $\tau_x = \{(i, x) : i \in I\} \in O$.

We now have all the ingredients to define what is a task. Note that this definition is independent of the communication model.

**Definition 22.** A task in a $n$-process system is a triple $(I, O, \Delta)$, where $I$ is the input complex, $O$ is the output complex, both of dimension $n-1$, and $\Delta : I \rightarrow 2^O$ is the input-output specification.

### A.4 Examples

For the readers more familiar with distributed graph problems (e.g., coloring, MIS, etc.) than with distributed system problems (e.g., consensus, renaming, etc.), let us define the task $(I, O, \Delta)$ of $k$-coloring the vertices of an $n$-node cycle $C_n$, with identifiers (IDs) in $[N] = \{1, \ldots, N\}$, $N \geq n$. A vertex of the input complex $I$ is a pair of the form $(i, (x, \{y, z\}))$, with $i \in [n]$, $\{x, y, z\} \subseteq [N]$, and $x \neq y \neq z \neq x$. The semantics of such a vertex is that process $i$ is handling some node of $C_n$ which received $x$ as ID, and the neighbors of this node in $C_n$ received IDs $y$ and $z$. A non-empty set $\{(i, (x_i, \{y_i, z_i\})) : i \in I\}$ with $I \subseteq [n]$ is a simplex of $I$ if the nodes of $C_n$ can be assigned distinct IDs in $[N]$ such that, for every $i \in I$, a node $u$ has ID $x_i$, and $\{y_i, z_i\}$ is the set of IDs of the two neighbors of $u$. A vertex of the output complex $O$ is a pair of the form $(i, (x, \{y, z\}, c))$, with $c \in [k] = \{1, \ldots, k\}$. Any non-empty set $\{(i, (x_i, \{y_i, z_i\}, c_i)) : i \in I\}$ of vertices forms a simplex of the output complex if the same condition as for the simplices of $I$ holds, and, in addition, for every $i \neq j \in I$, $c_i \neq c_j$ whenever $x_j \in \{y_i, z_i\}$ or $x_i \in \{y_j, z_j\}$. Let $\sigma = \{(i, (x_i, \{y_i, z_i\})) : i \in I\}$ in $I$. A simplex $\tau = \{(i, (x_i', \{y_i', z_i'\}, c_i)) : i \in I\}$ of $O$ satisfies $\tau \in \Delta(\sigma)$ if, for every $i \in I$, $(x_i', \{y_i', z_i'\}) = (x_i, \{y_i, z_i\})$.

Some models, e.g., the LOCAL model 33 [36], implicitly encode a network $G$ in the model itself. Every process $i$ is actually viewed as located at a node $i$ of $G$, which is exchanging information with the neighbors of node $i$ in $G$ only. In this case, the relevant task $(I', O', \Delta')$ is to $k$-color the vertices of the specific graph $G$ itself, with the additional constraints that the matching between the processes and the nodes of $G$ is fixed (but not known to the processes). Therefore, the vertices of $I'$ are merely defined as pairs $(i, x)$ with $x \in [N]$, and a non-empty set $\{(i, x_i) : i \in I\}$ with $I \subseteq [n]$ is a simplex of $I'$ if $x_i \neq x_j$ for every two indices $i, j \in I$. The output complex $O'$ has vertices $(i, c)$ with $c \in [k]$, and a non-empty set $\{(i, c_i) : i \in I\}$ with $I \subseteq [n]$ is a simplex of $O'$ if, for every $i, j \in I$, $c_j \neq c_i$ whenever $j \in N_G(i)$. For every $\sigma \in I$, $\Delta'(\sigma) = \{\tau \in O' : \text{name}(\tau) = \text{name}(\sigma)\}$.

These two variants of $k$-coloring, i.e., the tasks $(I, O, \Delta)$ and $(I', O', \Delta')$ above, are actually independent of the communication model. In particular, one can aim at solving the
task \((\mathcal{I}, \mathcal{O}, \Delta)\) for a graph \(G\) but in the \textsc{local} model with network \(H\), even if \(G\) and \(H\) are different. Conversely, one can aim at solving the task \((\mathcal{I}', \mathcal{O}', \Delta')\) in a communication model different from the \textsc{local} model with network \(G\).

Finally, observe that, for the task \((\mathcal{I}', \mathcal{O}', \Delta')\) to be non-trivial in the \textsc{local} model with network \(G\), it is required that the processes do not use their names for choosing their colors, as otherwise the processes could simply agree in advance on a specific \(k\)-coloring of \(G\). For instance, if \(G\) is the \(n\)-node path where every node \(i = 2, \ldots, n - 1\) is adjacent to nodes \(i + 1\) and \(i - 1\), even 2-coloring is trivial if processes can use their name: process \(i\) merely outputs \(i \mod 2\). To be relevant, the algorithm must therefore be name-independent. Such a constraint appears in various distributed computing settings, e.g., when aiming at solving the renaming task. We came back to the delicate issue names vs. identifiers in the section 3.

\section*{B Proof of Theorem 12}

Let us assume that there exists a \(t\)-round algorithm \(\text{alg}\) solving \((\mathcal{I}, \mathcal{O}, \Delta)\) in model \(\mathcal{M}\). This algorithm produces an output value \(\text{alg}(s)\) for every possible local state \(s\) of any process \(i\) after \(t\) rounds. By definition of the local state \(s\), the pair \((i, s)\) is a vertex of \(\mathcal{P}(t)\). Let us define the map \(\delta : \mathcal{P}(t) \rightarrow \mathcal{O}\) as

\[
\delta(i, s) \doteq (i, \text{alg}(s)).
\]

By definition, \(\delta\) is chromatic, and name-independent. Moreover, \(\text{alg}\) guarantees that, starting from any legal global input state, i.e., any simplex \(\sigma \in \mathcal{I}\), a legal global output state is produced, i.e., a simplex \(\tau \in \mathcal{O}\) is produced. The simplex is precisely the set \(\tau = \{(i, \text{alg}(s_i)) : i \in \text{name}(\sigma)\}\) where \(s_i\) is the local state of process \(i \in \text{name}(\sigma)\) after \(t\) rounds. It follows that \(\{\delta(i, s_i) : i \in \text{name}(\sigma)\} \subseteq \mathcal{O}\), and therefore \(\delta\) is simplicial. Let us now consider a closed simplex \(\sigma \in \mathcal{I}\), and an execution of \(\text{alg}\) in which all processes of \(\sigma\) communicate solely among themselves. By definition of \(\Xi\), after \(t\) rounds, every process \(i \in \text{name}(\sigma)\) ends up in a state \(s_i\) such that \(\{(i, s_i) : i \in \text{name}(\sigma)\}\) is a simplex of \(\Xi'(\sigma)\). \(\text{alg}\) then outputs \(\text{alg}(s_i)\) at every process \(i\). Since \(\text{alg}\) is correct, the resulting output simplex \(\tau = \{(i, \text{alg}(s_i)) : i \in \text{name}(\sigma)\}\) forms a global output state which is legal w.r.t. the global input state \(\sigma\). In other words, \(\tau \in \Delta(\sigma)\), from which it follows that \(\delta(\Xi'(\sigma)) \subseteq \Delta(\sigma)\), as desired.

Conversely, let us assume that there exists a chromatic name-independent simplicial map \(\delta : \mathcal{P}(t) \rightarrow \mathcal{O}\) such that, for every closed \(\sigma \in \mathcal{I}\), \(\delta(\Xi'(\sigma)) \subseteq \Delta(\sigma)\). Let us define the \(t\)-round algorithm \(\text{alg}\) as

\[
\text{alg}(s) \doteq \text{val}(\delta(i, s)),
\]

where \(\text{val}(i, x) = x\) for every vertex \((i, x)\) of any chromatic complex \((x\) is the value of that vertex). Since \(\delta\) is name-independent, \(\text{alg}\) is well defined. Moreover, since \(\delta\) is simplicial, the image \(\tau = \{(i, \text{alg}(s_i)) : i \in \text{name}(\sigma)\}\) of a global state \(\{(i, s_i) : i \in \text{name}(\sigma)\}\) of the system after \(t\) rounds starting from a global input state \(\sigma\) is a legal global output state. Moreover, since \(\delta(\Xi'(\sigma)) \subseteq \Delta(\sigma)\), if the processes in \(\text{name}(\sigma)\) have communicated solely among themselves, then \(\tau \in \Delta(\sigma)\), that is, \(\tau\) is a legal output state w.r.t. the input state \(\sigma\), as desired.

\section*{C Proof of Lemma 20}

Let us define the task \(\mathcal{T}' = (\mathcal{I}, \mathcal{O}', \Delta')\). Note that, for every \(i \in [n]\), the channels incoming to \(i\) are all the hyperedges \(e \in E_i\), where \(E_i = E_H(i)\) is the set of hyperedges of \(H\) containing \(i\).
For every $i \in [n]$, a pair $(i, K_i)$ is a vertex of $O'$ if $K_i$ is a node-labeled sub-hypergraph of $H$ induced by the processes in $J_i = \cup_{e \in E_i} e$ such that, if $t_{i,j} = (x_{i,j}, y_{i,j}) \in \text{val}(I) \times \text{val}(O)$ denotes the label of process $j \in J_i$, then

$$\{(j, y_{i,j}) : j \in J_i\} \in \Delta(\{(j, x_{i,j}) : j \in J_i\}).$$

Let $\sigma = \{(i, x'_i) : i \in I\}$ be a closed simplex of $I$. A set $\{(i, K_i) : i \in I\}$ of vertices of $O'$ is a simplex of $\Delta'(\sigma)$ if (1) $x_{i,i} = x'_i$ for every $i \in I$, and (2) for every $e \in E(\bar{H})$, and for every two processes $i, j \in I \cap e$, $\pi_e(K_i) = \pi_e(K_j)$.

We first show that $(I, O', \Delta')$ is edge-checkable. Let $\sigma = \{(i, x'_i) : i \in I\}$ be a closed simplex of $I$, and let $\tau = \{(i, K_i) : i \in I\}$. For one direction, let us assume that $\tau \in \Delta'(\sigma)$. We show that, for every $e \in E(H), \pi_e(\tau) \in \Delta'(\pi_e(\sigma))$. Let $e \in E(H)$. By definition of $\Delta'$ we have that, for every $e' \in E(H)$, for every $i, j \in e \cup e'$, $\pi_e(K_i) = \pi_e(K_j)$ and $x_{i,i} = x'_i$. It follows that $\pi_e(\tau) \in \Delta'(\pi_e(\sigma))$, as desired. For the reciprocal, let us assume that $\pi_e(\tau) \in \Delta'(\pi_e(\sigma))$ for every $e \in E(H)$. By definition of $\Delta'$, we have that, for every $i, j \in e$, $\pi_e(K_i) = \pi_e(K_j)$, and $x_{i,i} = x'_i$. This being true for all hyperedges in $H$, $\tau \in \Delta'(\sigma)$ holds, and thus $T'$ is indeed edge-checkable.

Second, we show how to construct a solution for $T'$ given any solution for $T$, in a single round. Let $\tau$ be a closed simplex of $O'$ in $\Delta(\sigma)$ for some closed simplex $\sigma \in I$. In one round, every process $i$ can collect the values and inputs of all the processes $j \in J_i = \cup_{e \in E_i} e$. By construction, the collection of resulting labeled sub-hypergraphs of $H$ induced by the processes in $J_i$, for $i \in \text{name}(\sigma)$, forms a valid solution in $\Delta'(\sigma)$.

Finally, we show how to construct a solution for $T$ given any solution for $T'$, in zero rounds. Let $\sigma = \{(i, x'_i) : i \in I\} \in I$ be a closed simplex, and let $\tau' = \{(i, K_i) : i \in I\} \in \Delta'(\sigma)$. We show that $\tau = \{(i, y_i) : i \in I\} \in \Delta(\sigma)$. Since $(I, O, \Delta)$ is locally checkable, it is sufficient to prove that $\pi_{J_i}(\tau) \in \Delta(\pi_{J_i}(\sigma))$ for every $i \in I$, with $J_i = \cup_{e \in E_i} e$. Let $i \in [n]$. We have $\pi_{J_i}(\tau') \in \text{Cl}(\Delta'(\sigma))$ merely because $\pi_{J_i}(\tau')$ is a face of $\tau'$. It follows that $\{\pi_{J_i}(K_i) : j \in e\} = \{\pi_e(K_i)\}$ for every $e \in E_i$. Therefore, for every process $j \in J_i$, $x_{i,j} = x_{j,j} = x'_j$ and $y_{i,j} = y_{j,j}$. In particular, $y_{i,j} = y_{j,j}$ implies that

$$\pi_{J_i}(\tau) = \{(j, y_{i,j}) : j \in J_i\}.$$  

By definition of the vertices in $O'$, we have $\{(j, y_{i,j}) : j \in J_i\} \in \Delta(\{(j, x_{i,j}) : j \in J_i\})$. Moreover $x_{i,j} = x'_j$. Therefore, $\pi_{J_i}(\tau) \in \Delta(\pi_{J_i}(\sigma))$ as desired, which concludes the proof.