We construct a time-optimal quasi-solution of the firing mob synchronisation problem over finite, connected, and undirected multigraphs whose maximum degrees are uniformly bounded by a constant. It is only a quasi-solution because its number of states depends on the graph or, from another perspective, does not depend on the graph but is countably infinite. To construct this quasi-solution we introduce signal machines over continuum representations of such multigraphs and construct a signal machine whose discretisation is a cellular automaton that quasi-solves the problem. This automaton uses a time-optimal solution of the firing squad synchronisation problem in dimension one with one general at one end to synchronise edges, and freezes and thaws the synchronisation of edges in such a way that all edges synchronise at the same time.

**Introduction.** The firing squad synchronisation problem in dimension one with one general at one end is to synchronise each finite one-dimensional array of cells starting from one end of the array and the cell at this end is called *general*. It was proposed by John R. Myhill in 1957, solved by John McCarthy and Marvin Lee Minsky, and published by Edward Forrest Moore in 1962 (see [Moo64]). The first
time-optimal several-thousand-states solution was found by Eiichi Goto in 1962 (see [Got62]), reduced to 16 states by Abraham Waksman in 1966 (see [Wak66]), and reduced to 8 states by Robert Balzer in 1967 (see [Bal67]). Hein D. Gerken found another time-optimal 7-states solution in 1987 (see [Ger87]) and Jacques Mazoyer found a time-optimal 6-states solution also in 1987 (see [Maz87]). It is unknown whether there is a time-optimal 5-states solution but it is known that there is no time-optimal 4-states solution, a result due to Robert Balzer and Peter Sanders (see [Bal76; San94]).

The firing mob synchronisation problem is to synchronise each finite, connected, and undirected graph whose maximum degree is bounded by a fixed constant starting from any vertex and this vertex is called *general*. It was solved by P. Rosenstiehl, J. R. Fiksel, and A. Holliger in 1972 (see [RFH72]) and also by Francesco Romani in 1976 (see [Ron76]), where the latter solution achieves better running times than the former. The problem for specific classes of graphs were for example studied by Kojiro Kobayashi in 1977 and 1978 (see [Kob77; Kob78a; Kob78b]) and by Zsuzsanna Róka in 2000 (see [Rók00]). Karel Culik II and Simant Dube presented a solution of the general case in 1991 (see [CD91]). It needs $3.5r$-many steps, where $r$ is the maximal distance of the general to a vertex and is called *radius of the graph with respect to the general*. By using more and more states, the solution can be adjusted such that the number of steps it needs approaches $3r$.

It was shown that $r + d$ is a lower bound for the number of steps that solutions of the firing mob synchronisation problem need by John J. Grefenstette in 1983 (see [Gre83]), where $d$ is the maximal distance between two vertices of the graph and is called *diameter of the graph*. Because there are graphs and choices of generals such that the diameter is $2r$, the solutions by Karel Culik II and Simant Dube approach the optimal number of steps, namely $3r$, if $r$ is taken as problem size. However, if $r + d$ is taken as problem size, then their solutions do not approach the optimal number of steps.

In the present chapter we construct a time-optimal quasi-solution that needs exactly $r + d$ steps but whose number of states depends on the graph or, from another perspective, does not depend on the graph but is countably infinite (this is why we call it a quasi-solution). It can also be turned into a time-optimal quasi-solution of the firing squad synchronisation problem for any region in any dimension with one general at any position by regarding each region to be synchronised as a graph, where cells in the region are vertices and edges are neighbourhood relationships.

However, restricted to specific classes of problems, the quasi-solution may not be time-optimal. For example, restricted to rectangular regions with one general at one corner, the quasi-solution needs $2(k + \ell - 2)$-many steps whereas $(k + \ell + \max\{k, \ell\} - 3)$-many steps is optimal (see for example [UYY09]), where $k$ and $\ell$ are the side lengths of the rect-
Nevertheless, because the quasi-solution is (trivially) embeddable in the sense of [Gre83], according to theorem 1 in [Gre83], it can be combined with finitely many embeddable time-optimal solutions for specific classes of problems to get one quasi-solution that is also time-optimal for those classes. Examples of solutions for specific classes, like rectangular regions with one general at the upper left corner, are given in sections 5 and 6 in [Gre83].

To design, explain, and draw solutions of firing squad/mob synchronisation problems, it is convenient to think about, talk about, and draw continuous space-time diagrams of different kinds of signals that move across the cell space, vanish or give rise to new signals upon reaching boundaries or junctions of the space or upon colliding with each other. This is mostly done in an informal way, but the idea of signals has also been formalised for one-dimensional cellular automata by Jérôme Olivier Durand-Lose in 2005 (see [Duro5]).

This formalisation however does not handle accumulations of events like collisions and does not allow infinitely many signals of different speeds, which naturally occur and are necessary in descriptions of many solutions of the firing squad synchronisation problem by signals. For example, collisions accumulate at the time synchronisation finishes and infinitely many signals of different speeds may originate from the general. In the time evolutions of the actual cellular automata, the accumulations of collisions disappear due to the discreteness of space and time, and the infinitely many signals are cleverly produced by finitely many states (see for example [Maz87]).

Because we want to describe our quasi-solution in terms of signals in a formal way, we first introduce continuum representations of finite and connected multigraphs (without self-loops), we secondly introduce signal machines over such representations that allow infinitely many signals of different speeds and seamlessly handle accumulations of events and accumulations of accumulations of events and so forth, and we thirdly construct a signal machine for the continuous firing mob synchronisation problem over such representations and shortly note how to discretise it to get a cellular automaton quasi-solution of the firing mob synchronisation problem.

**Contents.** In section 1 we state the firing squad and the firing mob synchronisation problems. In section 2 we introduce undirected multigraphs (without self-loops) and direction-preserving paths in such graphs, which are paths that do not make U-turns. In section 3 we introduce continuum representations of undirected multigraphs, which are in a sense drawings of graphs in a high-dimensional Euclidean space. In section 4 we introduce signal machines, which can be studied in their own right, but which can also be thought of as high-level views of time evolutions of cellular automata over graphs, like cellular automata over finitely right generated cell spaces, that are restricted to configurations...
with a fixed finite support. In section 5 we construct a signal machine whose discretisation is a cellular automaton that quasi-solves the firing mob synchronisation problem in \((r + d)\)-many steps. And in section 6 we sketch a proof for that statement. The impatient may right now have a look at the continuous space-time diagrams of the synchronisations of small trees as performed by the quasi-solution: See figures 5.2 to 5.5 on pages 32 to 35.

**Preliminary Notions.** The affinely extended real numbers \(\mathbb{R} \cup \{-\infty, +\infty\}\) are denoted by \(\mathbb{R}^\ast\). For each tuple \((r, r') \in \mathbb{R} \times \mathbb{R}\) such that \(r \leq r'\), the closed, open, and the two half-open extended real intervals with the endpoints \(r\) and \(r'\) are denoted by \([r, r']\), \((r, r']\), and \([r, r]\) and \([r, r')\) respectively. And, for each tuple \((z, z') \in \mathbb{Z} \times \mathbb{Z}\) such that \(z \leq z'\), the closed integer-valued interval with the endpoints \(z\) and \(z'\), namely \(\{z, z + 1, \ldots, z'\}\) or equivalently \([z, z'] \cap \mathbb{Z}\), is denoted by \([z : z']\).

1 THE FIRING SQUAD/MOB SYNCHRONISATION PROBLEMS

We formally state the problems for generalised cellular automata as introduced in [Wac16] over spaces as introduced in [Wac17]. However, you do not need to be familiar with these automata and spaces to understand the problems on an intuitive level or to understand the quasi-solutions in detail. Such an automaton is essentially a cellular automaton over a vertex-transitive graph, which means that each vertex of the graph is a copy of a fixed possibly-infinite-state machine, its inputs are the states of the vertex’s neighbours, and all vertices change their states synchronously.

In this section, let \(\mathcal{R} = ((M, G, \triangleright), (m_0, \{g_{m_0, m}\}_{m \in M}))\) be a finitely right generated cell space, let \(N\) be a finite right generating set of \(\mathcal{R}\) that contains \(G_0\), where \(G_0\) is the stabiliser of \(m_0\) under \(\triangleright\), let \(\mathcal{G} = (M, E)\) be the coloured \(N\)-Cayley graph of \(\mathcal{R}\), let \(\mathcal{C}\) be a semi-cellular automaton over \(\mathcal{R}\) with state set \(Q\), neighbourhood \(N\), and local transition function \(\delta\), and let \(\Delta\) be the global transition function of \(\mathcal{C}\).

To state the problems succinctly we introduce the notions of passive subsets of states, dead states, supports of global configurations with respect to a distinguished dead state, and what it means for a global configuration to be of the form of a pattern in the following four definitions.

**Definition 1.1.** Let \(P\) be a subset of \(Q\). It is called **passive** if and only if, for each local configuration \(\ell \in Q^N\) with \(\operatorname{im}(\ell) \subseteq P\), we have \(\delta(\ell) = \ell(G_0)\).

**Definition 1.2.** Let \(q\) be a state of \(Q\). It is called **dead** if and only if, for each local configuration \(\ell \in Q^N\) with \(\ell(G_0) = q\), we have \(\delta(\ell) = q\).
In the remainder of this section, let $Q$ contain a distinguished dead state named $\#$.

**Definition 3.5.** Let $c$ be a global configuration of $Q^M$. The set $\text{supp}(c) = M \setminus c^{-1}(\#)$ is called *support* of $c$. □

**Definition 3.6.** Let $A$ be a subset of $M$, let $p$ be a pattern of $Q^A$, and let $c$ be a global configuration of $Q^M$. The global configuration $c$ is said to be *of the form* $p$ if and only if there is an element $g \in G$ such that $\left|_{g \cdot A}^f c \right. = g \uparrow p$ and $\left|_{M \setminus (g \cdot A)}^f c \right. \equiv \#$. □

We state the firing squad synchronisation problem in

**Definition 3.7.** Let $\#, g, s$, and $f$ be four distinct states, and let $Q'$ be the set that consists of those states. A solution of the *firing squad synchronisation problem in dimension one with one general at the left end* is a cellular automaton $C$ over $((\mathbb{Z}, \mathbb{Z}, +), (0, \{ z \}_{z \in \mathbb{Z}})$ with neighbourhood $\{-1, 0, 1\}$ and finite set of states that includes $Q'$ such that the state $\#$ is dead and the set $\{\#, s\}$ is passive, and whose global transition function $\Delta$ has the following property:

For each global configuration $c$ with finite support of the form $gss \cdots s$, there is a non-negative integer $k$ such that the global configuration $\Delta^k(c)$ is of the form $ff \cdots f$ and has the same support as $c$, and such that the state $f$ does not occur in any of the global configurations $\Delta^j(c)$, for $j \in \mathbb{N}_0$ with $j < k$. □

**Remark 3.8.** Let $C$ be a solution of the above problem, let $c$ be a global configurations of the form $gss \cdots s$, and let $k$ be the non-negative integer from the problem definition. Then, because the state $\#$ is dead and the support of $\Delta^k(c)$ is the same as the one of $c$, for each non-negative integer $j$ with $j \leq k$, the support of $\Delta^j(c)$ is the same as the one of $c$. Broadly speaking, in the time evolution of solutions, the support of initial configurations can neither shrink nor grow before synchronisation is finished. Moreover, because the set $\{\#, s\}$ is passive, if the support of $c$ consists of at least 3 cells, then $\Delta(c)$ cannot be of the form $ff \cdots f$. Broadly speaking, the problem cannot be solved trivially. □

**Remark 3.9.** As mentioned above, for each global configuration $c$ of the form $gss \cdots s$, the supports of the global configurations that are observable in the time evolutions that begin in the configuration $c$ of cellular automata that solve the above problem, are included in the support of $c$. Hence, we can regard such cellular automata as automata over one-dimensional arrays with one dummy neighbour in the state $\#$ at each end. □
Remark 1.8. The above problem can be generalised in many ways. For example, by allowing the general to be placed anywhere or by allowing more than one general.

We state the firing mob synchronisation problem in

**Definition 1.9.** Let \( \#, g, s, \) and \( f \) be four distinct states, and let \( Q' \) be the set that consists of those states. A solution of the firing mob synchronisation problem in \( \mathcal{R} \) with respect to \( S \) is a semi-cellular automaton over \( \mathcal{R} \) with neighbourhood \( S \) and finite set of states that includes \( Q' \) such that the state \( \# \) is dead and the set \( \{\#, s\} \) is passive, and whose global transition function \( \Delta \) has the following property:

For each finite subset \( A \) of \( M \) such that the subgraph \( G[A] \) of \( G \) induced by \( A \) is connected, each element \( a \in A \), each pattern \( p \in Q^A \) such that \( p(a) = g \) and \( p|_{A \setminus \{a\}} \equiv s \), and each global configuration \( c \) of the form \( p \), there is a non-negative integer \( k \) such that the global configuration \( \Delta^k(c) \) is of the form \( A \to Q, a \mapsto f \), and such that the state \( f \) does not occur in any of the global configurations \( \Delta^j(c) \), for \( j \in \mathbb{N}_0 \) with \( j < k \).

Remark 1.10. The firing squad synchronisation problem with one general at an arbitrary position is the firing mob synchronisation problem in \( \mathcal{R} \) with respect to \( S \) is a semi-cellular automaton over \( \mathcal{R} \) with neighbourhood \( S \) and finite set of states that includes \( Q' \) such that the state \( \# \) is dead and the set \( \{\#, s\} \) is passive, and whose global transition function \( \Delta \) has the following property:

For each finite subset \( A \) of \( M \) such that the subgraph \( G[A] \) of \( G \) induced by \( A \) is connected, each element \( a \in A \), each pattern \( p \in Q^A \) such that \( p(a) = g \) and \( p|_{A \setminus \{a\}} \equiv s \), and each global configuration \( c \) of the form \( p \), there is a non-negative integer \( k \) such that the global configuration \( \Delta^k(c) \) is of the form \( A \to Q, a \mapsto f \), and such that the state \( f \) does not occur in any of the global configurations \( \Delta^j(c) \), for \( j \in \mathbb{N}_0 \) with \( j < k \).

Remark 1.11. Each semi-cellular automaton over \( \mathcal{R} \) with neighbourhood \( S \) is equivalent to a cellular automaton over the coloured \( S \)-Cayley graph of \( \mathcal{R} \) acted upon by its automorphism group, in the sense that, for each of the former kind of automata, there is one of the latter kind with the same global transition function, and vice versa. Note that the stabilisers of coloured \( S \)-Cayley graphs of \( \mathcal{R} \) are trivial, and hence the notions of semi-cellular and cellular automata are identical over such graphs.

Remark 1.12. We can regard semi-cellular automata that solve the above problem as semi-cellular automata over subgraphs of \( G \) that are induced by finite subsets of \( M \) with one dummy neighbour in the state \( \# \) at each edge that leads out of the subgraph. Note that, because the graph \( G \) is of bounded degree, the maximum degrees of the subgraphs it induces are uniformly bounded by a constant.

Remark 1.13. Ideally we would like an abstract description of a semi-cellular automaton that does not depend on any specifics of \( \mathcal{R} \) and \( S \) and that yields a solution for each choice of \( \mathcal{R} \) and \( S \) or at least for as huge a class of such choices as possible.
Undirected multigraphs without self-loops are introduced in

**Definition 2.1.** Let $V$ and $E$ be two disjoint sets, and let $\varepsilon$ be a map from $E$ to $\{\{v, v'\} \subseteq V \mid v \neq v'\}$. The triple $\mathcal{G} = (V, E, \varepsilon)$ is called undirected multigraph; each element $v \in V$ is called vertex; each element $e \in E$ is called edge; and, for each edge $e \in E$, each vertex of $\varepsilon(e)$ is called end of $e$.

**Remark 2.2.** Because each set in the codomain of $\varepsilon$ consists of exactly two distinct vertices, there are no self-loops in the undirected multigraph $\mathcal{G}$. With minor modifications the theory and the automata presented in this chapter also work if there are self-loops. They were merely excluded to make the presentation a little simpler.

In the remainder of this section, let $\mathcal{G} = (V, E, \varepsilon)$ be an undirected multigraph.

What being finite means for multigraphs is said in

**Definition 2.3.** The multigraph $\mathcal{G}$ is called finite if and only if the sets $V$ and $E$ are both finite.

Isolated vertices are the ones without incident edges as introduced in

**Definition 2.4.** Let $v$ be a vertex of $\mathcal{G}$. It is called isolated if and only if, for each edge $e \in E$, we have $v \notin \varepsilon(e)$.

Directed edges are edges with distinguished source and target vertices as introduced in

**Definition 2.5.** Let $e$ be an edge of $\mathcal{G}$, and let $v_1$ and $v_2$ be two vertices of $\mathcal{G}$ such that $\{v_1, v_2\} = \varepsilon(e)$. The triple $\vec{e} = (v_1, e, v_2)$ is called directed edge from $v_1$ through $e$ to $v_2$; the vertex $\sigma(\vec{e}) = v_1$ is called source of $\vec{e}$; the edge $\beta(\vec{e}) = e$ is called bed of $\vec{e}$; and the vertex $\tau(\vec{e}) = v_2$ is called target of $\vec{e}$.

At each vertex there is an empty path that starts and ends at the vertex, and non-empty paths are concatenations of directed edges with matching source and target vertices as introduced in

**Definition 2.6.** Let $v$ be a vertex of $\mathcal{G}$. The singleton $p = (v)$ is called empty path in $v$, the vertex $\sigma(p) = \tau(p) = v$ is called source and target of $p$, and the non-negative integer $|p| = 0$ is called length of $p$. 

\begin{align*}
\text{undirected multigraph} \\
\mathcal{G} = (V, E, \varepsilon) \\
\text{vertex} v \\
\text{edge} e \\
\text{ends} \varepsilon(e) \text{ of } e \\
\text{finite multigraph} \\
\text{isolated vertex} \\
\text{directed edge} \vec{e} \\
\text{from } v_1 \text{ through } e \text{ to } v_2 \\
\text{source} \sigma(\vec{e}) \text{ of } \vec{e} \\
\text{bed} \beta(\vec{e}) \text{ of } \vec{e} \\
\text{target} \tau(\vec{e}) \text{ of } \vec{e} \\
\text{empty path} (v) \text{ in } v \\
\text{source} \sigma((v)) \text{ and } \text{target} \tau((v)) \text{ of } (v) \\
\text{length} |(v)| \text{ of } (v)
\end{align*}
B. Let $n$ be a positive integer and, for each index $i \in [1 : n]$, let $\vec{e}_i$ be a directed edge of $\mathcal{G}$ such that, if $i \neq 1$, then $\sigma(\vec{e}_i) = \tau(\vec{e}_{i-1})$. The $(2n + 1)$-tuple $p = (\sigma(\vec{e}_1), \beta(\vec{e}_1), \tau(\vec{e}_1), \ldots, \beta(\vec{e}_n), \tau(\vec{e}_n))$ is called path from $\sigma(\vec{e}_1)$ to $\tau(\vec{e}_n)$; the vertex $\sigma(p) = \sigma(\vec{e}_1)$ is called source of $p$; the vertex $\tau(p) = \tau(\vec{e}_n)$ is called target of $p$; and the positive integer $|p| = n$ is called length of $p$.

C. The set of paths is denoted by $P$.

**Remark 2.7.** Each directed edge is a path of length 1.

Subpaths are connected parts of paths as introduced in

**Definition 2.8.** Let $p = (v_0, e_1, v_1, \ldots, e_n, v_n)$ be a path of $\mathcal{G}$, and let $k$ and $\ell$ be two indices of $[0 : n]$ such that $k \leq \ell$. The path $(v_k)$, if $k = \ell$, or $(v_k, e_{k+1}, v_{k+1}, \ldots, e_\ell, v_\ell)$, otherwise, is called subpath of $p$ and is denoted by $p_{[k,\ell]}$.

Direction-preserving paths are the ones without U-turns as introduced in

**Definition 2.9.** Let $p = (v_0, e_1, v_1, \ldots, e_n, v_n)$ be a path of $\mathcal{G}$. It is called direction-preserving if and only if, for each index $i \in [1 : n-1]$, we have $e_i \neq e_{i+1}$. The set of direction-preserving paths is denoted by $P_\rightarrow$.

Two paths with matching target and source vertices can be concatenated as introduced in

**Definition 2.10.** Let $p = (v_0, e_1, v_1, \ldots, e_n, v_n)$ and $p' = (v'_0, e'_1, v'_1, \ldots, e'_n, v'_n)$ be two paths of $\mathcal{G}$ such that $v_n = v'_0$. The path $p \bullet p' = (v_0, e_1, v_1, \ldots, e_n, v_n, e'_1, v'_1, \ldots, e'_n, v'_n)$ is called concatenation $p \bullet p'$ of $p$ and $p'$.

**Remark 2.11.** Each non-empty path is the concatenation of directed edges.

What being connected means for multigraphs is said in

**Definition 2.12.** The multigraph $\mathcal{G}$ is called connected if and only if, for each tuple $(v, v') \in V \times V$, there is a path from $v$ to $v'$.

**Remark 2.13.** The multigraph $\mathcal{G}$ is connected if and only if, for each tuple $(v, v') \in V \times V$, there is a direction-preserving path from $v$ to $v'$.

We can assign weights to edges as done in
Definition 2.14. Let \( \omega \) be a map from \( E \) to \( \mathbb{R}_{>0} \). It is called edge weighting of \( G \), and, for each edge \( e \in E \), the element \( \omega(e) \) is called edge weight of \( e \).

Edge weights induce weights of paths as introduced in Definition 4.37.

Definition 2.15. Let \( \omega \) be an edge weighting of \( G \) and let \( p = (v_0, e_1, v_1, \ldots, e_n, v_n) \) be a path of \( G \). The sum \( \omega(p) = \sum_{i=1}^{n} \omega(e_i) \) is called weight of \( p \).

Remark 2.16. Each empty path has weight 0.

Remark 2.17. Each directed edge has the same weight as its bed.

3 CONTINUUM REPRESENTATION

In this section, let \( G = (V, E, \varepsilon) \) be an undirected multigraph without isolated vertices and let \( \omega \) be an edge weighting of \( G \). An orientation is a choice of source and target vertices for each edge as introduced in

Definition 3.1. Let \( \sigma \) and \( \tau \) be two maps from \( E \) to \( V \) such that, for each edge \( e \in E \), we have \( \{\sigma(e), \tau(e)\} = \varepsilon(e) \). The tuple \( (\sigma, \tau) \) is called orientation of \( G \).

Realising weighted edges as disjoint intervals and gluing these intervals together at shared ends yields a continuum representation of \( G \) and is done in

Definition 3.2. Let \( (\sigma, \tau) \) be an orientation of \( G \), let

\[
\zeta : \mathbb{R} \setminus \{0\} \to \{\sigma, \tau\},
\]

\[
r \mapsto \begin{cases} \sigma, & \text{if } r < 0, \\ \tau, & \text{if } r > 0, \end{cases}
\]

let

\[
\begin{align*}
\phi : E &\to \mathbb{R}_{<0}, \\
e &\mapsto -\frac{\omega(e)}{2},
\end{align*}
\]

and

\[
\begin{align*}
\psi : E &\to \mathbb{R}_{>0}, \\
e &\mapsto \frac{\omega(e)}{2},
\end{align*}
\]

and let \( \sim \) be the equivalence relation on \( \mathbb{R} \times E \) such that, for each tuple \( (r, e) \in \mathbb{R} \times E \) and each tuple \( (r', e') \in \mathbb{R} \times E \),

\[
(r, e) \sim (r', e') \iff r \in \{\phi(e), \psi(e)\} \\
\wedge r' \in \{\phi(e'), \psi(e')\} \\
\wedge \zeta(r)(e) = \zeta(r')(e').
\]

The set \( \hat{G} = (\bigcup_{e \in E}[\phi(e), \psi(e)] \times \{e\})/\sim \) is called continuum representation of \( G \).
Remark 3.3. Each weighted edge is realised as a closed interval whose length is the edge’s weight. These intervals are made disjoint by taking the Cartesian product with the respective edge. And they are glued together at shared ends by taking the set of all these disjoint intervals modulo the equivalence relation $\sim$. The vertices are implicitly realised as end points or junctions of the glued disjoint intervals. □

Remark 3.4. If the graph $\mathcal{G}$ contained isolated vertices, then they would not be represented in $\mathcal{G}$. With minor modifications the theory presented in this chapter also works if there are isolated vertices. They were merely excluded to make the presentation a little simpler. □

In the remainder of this section, let $\mathcal{G}$ be a continuum representation of $\mathcal{G}$ with respect to an orientation $(\sigma, \tau)$, and let $\zeta$, $\phi$, $\psi$, and $\sim$ be the maps and the equivalence relation from definition 3.2.

Vertices are canonically embedded into $\mathcal{G}$ as is done in

**Definition 3.5.** The map

\[
\begin{align*}
\gamma &: V \to \mathcal{G}, \\
\sigma(e) &\mapsto [(\phi(e), e)]_\sim, \\
\tau(e) &\mapsto [(\psi(e), e)]_\sim,
\end{align*}
\]

embeds vertices of $\mathcal{G}$ into $\mathcal{G}$. Its image is denoted by $V$ and each element $v \in V$ is called **vertex**.

**Remark 3.6.** The embedding is well-defined due to the definition of the equivalence relation $\sim$. □

Edges are canonically embedded into the power set of $\mathcal{G}$ as is done in

**Definition 3.7.** The map

\[
\begin{align*}
\gamma &: E \to \mathcal{P}(\mathcal{G}), \\
e &\mapsto ([\phi(e), \psi(e)] \times \{e\})/\sim,
\end{align*}
\]

embeds edges of $\mathcal{G}$ into $\mathcal{G}$. Its image is denoted by $E$ and each element $e \in E$ is called **edge**.

**Remark 3.6.** The embedding is well-defined due to the definition of the equivalence relation $\sim$. □

At each point of $\mathcal{G}$ there is at least one direction to move: In a vertex of degree $k$, there are $k$ directions; and on an edge but not in one of its endpoints, there are 2 directions. An inefficient but immediate way to represent these directions is as in

**Definition 3.8.** The set $\{-1, 1\} \times E$ is denoted by $\text{Dir}$, each element $d = (o, e) \in \text{Dir}$ is called **direction on e**, the element $o$ is called **orient-**
ation of $d$, the involution
\[ - : \text{Dir} \to \text{Dir}, \quad (o, e) \mapsto (o, e), \]
reverses the orientation of directions, and the map
\[ \text{dir} : \mathcal{G} \to \mathcal{P}(\text{Dir}), \]
\[ [(r, e)] \mapsto \begin{cases} \{(-1, e), (1, e)\}, & \text{if } r \in \phi(e), \psi(e) \in \mathbb{R}_\geq 0, \\ \{(-\text{sgn}(r'), e') \mid (r', e') \in [(r, e)]\}, & \text{otherwise}, \end{cases} \]
assigns to each point in $\mathcal{G}$ the set of possible directions in which someone standing on that point can move.

\[ \square \]

**Remark 3.9.** This representation of directions is inefficient in the following sense: If we stand on an edge but not on one of its endpoints, then the orientation is enough directional information; and if we stand on a vertex, then the edge is enough directional information, because the orientation is implicit in the fact that we can only move onto the edge but not off it since we are in one end of the edge. On a vertex we do not even need the edge itself but only an identifier for the edge that is locally unique; for example, we could colour the edges such that no two edges of the same colour are incident to the same vertex and use this colour instead.

\[ \square \]

Like vertices, paths of $\mathcal{G}$ are also canonically embedded into $\mathcal{G}$ and each embedding can be unit-speed parametrised by the interval from 0 to the path’s weight as is inductively done in

**Definition 3.10.** The map
\[ - : \mathcal{P} \to \mathcal{G}\{(0, r) | r \in \mathbb{R}_\geq 0\}, \]
\[ (v) \mapsto \begin{cases} [0, 0] \to \mathcal{G}, \\ r \mapsto v \end{cases}, \]
\[ (\sigma(e), e, \tau(e)) \mapsto \begin{cases} [0, \omega(e)] \to \mathcal{G}, \\ r \mapsto ([\phi(e) + r, e)]_{\sim}, \end{cases} \]
\[ (\tau(e), e, \sigma(e)) \mapsto \begin{cases} [0, \omega(e)] \to \mathcal{G}, \\ r \mapsto ([\psi(e) - r, e)]_{\sim}, \end{cases} \]
\[ (v_0, e_1, v_1) \bullet p' \mapsto \begin{cases} [0, \omega((v_0, e_1, v_1) \bullet p')] \to \mathcal{G}, \\ r \mapsto \begin{cases} (v_0, e_1, v_1)(r), & \text{if } r \leq \omega(e_1), \\ \gamma(r - \omega(e_1)), & \text{otherwise}, \end{cases} \end{cases} \]
maps paths of $\mathcal{G}$ to unit-speed parametrisations of them in $\mathcal{G}$.

\[ \square \]

**Remark 3.11.** The base cases of the inductive definition do not overlap because there are no self-loops, and the inductive step is well-defined because $\omega((v_0, e_1, v_1) \bullet p') = \omega(e_1) + \omega(p')$.

\[ \square \]
The images $\mathcal{P}$ and $\mathcal{P}_-$ consist broadly speaking of paths and direction-preserving paths in $\mathfrak{G}$ from vertices to vertices that only change direction at vertices. Restricting the parametrisation intervals of paths in $\mathcal{P}_-$ to subintervals and doing a reparametrisation such that the new parametrisation starts at 0 yields all direction-preserving paths in $\mathfrak{G}$ and is done in

**Definition 3.12.** The set

$$\{p|_{[r,s]}(-r) \mid p \in \mathcal{P}_- \text{ and } r, s \in [0, \omega(p)] \text{ with } r \leq s\}$$

is denoted by $\mathcal{P}_-$; each element $p \in \mathcal{P}_-$ is called direction-preserving path, the length of the interval $\text{dom}(p)$ is called length of $p$ and is denoted by $\omega(p)$, the point $\sigma(p) = p(0)$ is called source of $p$, the point $\tau(p) = p(\omega(p))$ is called target of $p$, and the path $p$ is called empty if and only if $\omega(p) = 0$.

**Remark 3.35.** Doing the same with the paths in $\mathcal{P}$ does not yield all paths in $\mathfrak{G}$ but only those that change direction at vertices and not on edges. Because we only need direction-preserving paths in what is to come, we do not define what a general path on $\mathfrak{G}$ is.

**Remark 3.36.** Sources and targets of direction-preserving paths in $\mathfrak{G}$ are in general not vertices.

The distance between two points is the length of the shortest path between the points as introduced in

**Definition 3.15.** The map

$$d: \mathfrak{G} \times \mathfrak{G} \to \mathbb{R}_{\geq 0} \cup \{\infty\},$$

$$(m, m') \mapsto \inf\{\omega(p) \mid p \in \mathcal{P}_- \text{ with } \sigma(p) = m \text{ and } \tau(p) = m'\}$$

is called distance, where the infimum of the empty set is infinity.

**Remark 3.16.** If the graph $\mathcal{G}$ is finite and connected, then the distance map $d$ is a metric. Otherwise, it may not be a metric. For example, if there are two distinct vertices $v, v' \in V$ such that, for each $n \in \mathbb{N}_+$, there is an edge $e \in E$ whose weight is $1/n$, then the distance of $v$ and $v'$ is 0 although $v \neq v'$. Or, if the graph $\mathcal{G}$ is not connected, then there are two points $m, m' \in \mathfrak{G}$ whose distance is $\infty$.

Each non-zero vector of a vector space is uniquely determined by its magnitude and its direction, and the zero vector is already uniquely determined by its magnitude, which is 0, and can be thought of as pointing in every direction, which can be represented by the set of directions. A generalisation of vector spaces is given in
Definition 3.17. Let \( vry \) be the set Dir. The set
\[
\text{Arr} = \{(0, vry)\} \cup (\mathbb{R}_{>0} \times \text{Dir})
\]
is called \textit{arrow space}; each element \( a \in \text{Arr} \) is called \textit{arrow}; the set \( vry \) is called \textit{semi-direction}; for each element \( a = (r, d) \in \text{Arr} \), the real number \( \|a\| = r \) is called \textit{magnitude} of \( a \), and the (semi-)direction \( \text{dir}(a) = d \) is called (semi-)\textit{direction} of \( a \).

Arrow spaces will be used to represent both velocities, which are directed speeds, and directed distances. Multiplying a velocity by a time yields a directed distance. This scalar multiplication is introduced in

Definition 3.18. The map
\[
\cdot : \text{Arr} \times \mathbb{R}_{\geq 0} \to \text{Arr},
\]
\[
(r, d), s \mapsto \begin{cases} 
(0, vry), & \text{if } s = 0, \\
(r \cdot s, d), & \text{if } s > 0,
\end{cases}
\]
is called \textit{scalar multiplication}.

When we stand at the beginning of a non-empty direction-preserving path and walk along it until we reach its end, we start our walk on the first edge of the path in a certain direction and we end it on the last edge of the path in a certain direction. These directions are introduced in

Definition 3.19. Let \( p \) be a direction-preserving path of \( \mathcal{P} \). If \( p \) is empty, let \( \text{dir}_\sigma(p) = vry \) and let \( \text{dir}_\tau(p) = vry \).

Otherwise, there are two edges \( e_\sigma, e_\tau \in E \), which may be the same, and there are two positive real numbers \( \xi_\sigma, \xi_\tau \in ]0, \omega(p)[ \) such that \( p([0, \xi_\sigma]) \subseteq e_\sigma \) and \( p([\omega(p) - \xi_\tau, \omega(p)]) \subseteq e_\tau \). Moreover, there are four real numbers \( r_\sigma, r'_\sigma, r_\tau, r'_\tau \) such that \( [(r_\sigma, e_\sigma)],_\sigma = p([0, \xi_\sigma]) \), \( [(r'_\sigma, e_\sigma)],_\sigma = p(\omega(p) - \xi_\sigma) \), and \( [(r'_\sigma, e_\sigma)],_\sigma = p(\omega(p)) \). Let \( \text{dir}_\sigma(p) = (\text{sgn}(r'_\sigma - r_\sigma), e_\sigma) \) and let \( \text{dir}_\tau(p) = (\text{sgn}(r'_\tau - r_\tau), e_\tau) \).

In both cases, the (semi-)direction \( \text{dir}_\sigma(p) \) is called \textit{source direction} of \( p \) and the (semi-)direction \( \text{dir}_\tau(p) \) is called \textit{target direction} of \( p \). □

Remark 3.20. The edge \( e_\sigma \) is the first edge of the path \( p \) and the edge \( e_\tau \) is its last edge. The numbers \( \xi_\sigma \) and \( \xi_\tau \) are two positive real numbers such that the first \( \xi_\sigma \) length units of the path run on its first edge and the last \( \xi_\tau \) length units of the path run on its last edge. The numbers \( r_\sigma \) and \( r'_\sigma \) are the positions of the path on its first edge at its very beginning and after \( \xi_\sigma \) length units, and the numbers \( r_\tau \) and \( r'_\tau \) are the positions of the path on its last edge \( \xi_\tau \) length units before its end and at its very end. Therefore, the signum of \( r'_\sigma - r_\sigma \) is the start
direction on the first edge of the path and the signum of \( r' - r \) is the end direction on the last edge of the path.

\( \square \)

**Remark 3.21.** For each non-empty path \( p \in \mathcal{P}_\ast \), we have \( \text{dir}_\sigma(p) \in \text{dir}(\sigma(p)) \) and \( \text{dir}_\tau(p) \in -\text{dir}(\tau(p)) \).

\( \square \)

## 4 Signal Machines

In this section, let \( G = (V, E, \varepsilon) \) be a non-trivial, finite, and connected undirected multigraph, let \( \omega \) be an edge weighting of \( G \), and let \( M \) be a continuum representation of \( G \). Recall that, according to our definition of undirected multigraphs, there are no self-loops in \( G \). To motivate the definitions in this section, we talk as if there were a signal machine in front of us whose time evolution we can observe, although this evolution is not completely defined until the end of this section.

If you observe the time evolution of a signal machine on the graph \( M \), you see signals of different kinds and various speeds each carrying some data move along edges. When signals collide, they may be reflected, removed, new signals may be created, and so on. Similarly, when signals reach a vertex, they may be removed, copies of them may be sent onto all incident edges, new signals may be created, and so on. You may also see stationary signals and signals that travel side-by-side at the same speed. What happens when signals collide or reach a vertex is decided by two local transition functions, one that handles such events in vertices and one that handles them on edges.

The only events on edges are collisions. In each collision on an edge, there are at least two signals involved, the involved signals are either stationary or they move in one of the two possible directions, and at least two of the signals collide head-on or rear-end. Such a collision results in a set of signals that are either stationary or move away from the vertex along incident edges.

A vertex may be reached by just one signal or multiple signals may collide in it. In both cases, there is at least one signal involved, the involved signals are either stationary or they move towards the vertex just before the event, and at least one signal is moving. Such an event results in a set of signals that are either stationary or move away from the vertex along incident edges.

**Definition 4.1.** Let \( \text{Knd} \) be a set, let \( \text{spd} \) be a map from \( \text{Knd} \) to \( \mathbb{R}_{\geq 0} \), let \( \{D_k\}_{k \in \text{Knd}} \) be a family of sets, let

\[
\text{Sgnl} = \{(k, d, u) \mid k \in \text{Knd}, (\text{spd}(k), d) \in \text{Arr}, \text{ and } u \in D_k\},
\]

let \( \text{Dir}_e = \{\{d, -d\} \mid d \in \text{Dir}\} \), let

\[
\text{dom}(\delta_e) = \{S \in \mathcal{P}(\text{Sgnl}) \mid \exists D \in \text{Dir}_e : |S| \geq 2 \text{ and } \forall (k, d, u) \in S : d \in \{\text{vry}\} \cup D \text{ and}
\]

\[\]
let $\delta_e$ be a map from $\text{dom}(\delta_e)$ to $\mathcal{P}(\text{Sgnl})$ such that
\[
\forall S \in \text{dom}(\delta_e) \exists D \in \text{Dir}_e : \forall (k, d, u) \in S : d \in \{\text{vry}\} \cup D,
\]
and let $\delta_v$ be a map from $\text{dom}(\delta_v)$ to $\mathcal{P}(\text{Sgnl})$ such that
\[
\forall (D, S) \in \text{dom}(\delta_v) \forall (k, d, u) \in \delta_v(D, S) : d \in \{\text{vry}\} \cup D.
\]

The quadruple $\mathcal{S} = (\text{Knd}, \text{spd}, \{\text{Dt}_k\}_{k \in \text{Knd}}, (\delta_e, \delta_v))$ is called signal machine; each element $k \in \text{Knd}$ is called kind; for each kind $k \in \text{Knd}$, the non-negative real number $\text{spd}(k)$ is called speed of $k$ and the set $\text{Dt}_k$ is called data set of $k$; each element $s \in \text{Sgnl}$ is called signal; and the maps $\delta_e$ and $\delta_v$ are called local transition function on edges and in vertices respectively.

\[\square\]

**Remark 4.2.** The local transition function $\delta_e$ is used to handle events on edges but not in their endpoints. It gets the signals that are involved in the event and returns the resulting signals. Because in each event at least one moving signal is involved, the direction of this signal and the map that reverses orientation can be used by $\delta_e$ to determine the two possible directions the resulting signals may have.

The local transition function $\delta_v$ is used to handle events in vertices. It gets the directions signals may take at the respective vertex and the signals that are involved in the event and returns the resulting signals.

The local transition functions $\delta_e$ and $\delta_v$ are supposed to regard directions as black boxes that can merely be distinguished and whose orientation can be reversed. They must not determine edges or vertices by deconstructing directions, which is possible with the chosen representation of directions. If they did something like that, the signal machine would not be uniform.

\[\square\]

**Remark 4.3.** At the beginning of this section we fixed a general multigraph. This multigraph should be regarded as the blueprint of a multigraph. A signal machine depends only on that blueprint and not on any specific properties that a concrete choice of a multigraph may have. So, one and the same signal machine can be instantiated for any multigraph, each instantiation results in a machine on a concrete multigraph, and these instantiations are uniform in the chosen multigraphs. In other words, a signal machine is a map from the set of all multigraphs to the...
set of quadruples that describe instantiations of the machine on concrete multigraphs and this map depends only in a trivial way on its argument.

The quadruple that describes signal machines could be made independent of multigraphs by choosing a different representation for directions. They could for example be represented by integers or vectors or colours equipped with an involution to switch the orientation of directions.

Even the global transition function, which is introduced below and describes the time evolution of a signal machine, could be made independent of multigraphs by representing them as patterns in high-dimensional Euclidean spaces (think of the drawing of a graph on a piece of paper). The directions are then vectors that are tangential to edges.

In classical solutions of the firing squad synchronisation problem the regions to be synchronised are actually represented as patterns in integer lattices: The cells outside the region are in the same state, say 0, and all cells inside the region are not in state 0, more precisely, one cell inside the region is in a state that distinguishes it as the general, say 1, and the other cells inside the region are all in the same state, say 2.  

In the remainder of this section, let \( S = (K_{nd}, spd, \{D_{tk}\}_{k \in K_{nd}}, (\delta_{e}, \delta_{v})) \) be a signal machine.

To describe the time evolution of our signal machine, the following notions are convenient.

**Definition 6.**

Let \( T \) be the set \( \mathbb{R}_{\geq 0} \), let \( \hat{T} \) be the set \( T \cup \{\infty\} \), let \( Q \) be the set \( \mathcal{P}(\text{Sgnl}) \), and let \( \text{Cnf} \) be the set \( Q^{M} \). Each element \( t \in T \) is called time and the time \( \infty \) is called improper, each element \( q \in Q \) is called state, and each element \( c \in \text{Cnf} \) is called configuration.

The components of signals and some compounds of them are named in

**Definition 4.4.** Let \( T \) be the set \( \mathbb{R}_{\geq 0} \), let \( \hat{T} \) be the set \( T \cup \{\infty\} \), let \( Q \) be the set \( \mathcal{P}(\text{Sgnl}) \), and let \( \text{Cnf} \) be the set \( Q^{M} \). Each element \( t \in T \) is called time and the time \( \infty \) is called improper, each element \( q \in Q \) is called state, and each element \( c \in \text{Cnf} \) is called configuration.  

**Remark 6.** For each time \( t \in T \), the arrow \( \text{vel}(s) \cdot t \) is equal to the arrow \( (\text{spd}(k) \cdot t, d) \), which can be interpreted as a directed distance.

Signals of speed 0 are named in

**Definition 4.7.** Let \( s \) be a signal of \( \text{Sgnl} \). It is called stationary if and only if \( \text{spd}(s) = 0 \).
When we stand on a point facing in a direction and from there we
walk a fixed distance without making U-turns but otherwise making
arbitrary choices at each vertex, then there is a finite number of points
that we may reach and we reach them walking in some direction. The
set of these points with and without target-directions is introduced in

**Definition 4.8.** Let \( m \) be a point of \( M \) and let \((\ell, d)\) be an arrow.
The set of points with target-directions that can be reached from \( m \) by
a direction-preserving path of length \( \ell \) with source-direction \( d \) is

\[
R_{(\ell,d)}^{-}(m) = \{ (\tau(p), \text{dir}_\tau(p)) \mid p \in \mathcal{P}_-, \omega(p) = \ell, \\
\text{dir}_\sigma(p) = d, \text{ and } \sigma(p) = m \}
\]

and without target-directions it is

\[
R_{(\ell,d)}(m) = \{ m' \mid \exists d \in \text{Dir} : (m', d) \in R_{(\ell,d)}^{-}(m) \}.
\]

An event occurs when a signal reaches a vertex or two signals coming
from different points collide. The time of the next event is given a name
in

**Definition 4.9.** Let \( c \) be a configuration of \( \text{Cnf} \). The minimum time
until a signal in \( c \) reaches a vertex is

\[
t' = \inf_{m \in M} \inf_{s \in c(m)} \inf_{\text{sp}(s) > 0} \{ t \in \mathbb{R} > 0 \mid R_{\text{vel}(s), t}(m) \cap \mathcal{V} \neq \emptyset \}.
\]

The minimum time until at least two signals in \( c \) collide is

\[
t'' = \inf_{m, m' \in M} \inf_{s \in c(m)} \inf_{m' \neq m'} \inf_{s' \in c(m') \text{ }} \{ t \in \mathbb{R} > 0 \mid R_{\text{vel}(s), t}(m) \cap R_{\text{vel}(s'), t}(m') \neq \emptyset \}.
\]

The minimum time until the next event(s) in \( c \) occurs is

\[
t_0(c) = \min\{t', t''\}.
\]

**Remark 4.10.** A stationary signal at a vertex does never reach the
vertex (it is already there) and two signals that already are at the
same vertex do never collide (they may have collided when they got
there but now they just are at the same vertex; if they have a non-zero
velocity, then they will leave the vertex without interfering each
other, and, if they have the same positive velocity, then they will travel
alongside each other).

**Remark 4.11.** The next event time may be 0, which means that events
accumulate at time 0, or \( \infty \), which means that there are no more events
in the future; note that \( \inf(\emptyset) = \infty \). It is for example 0 if there is a
sequence of signals moving at the same velocity towards the same vertex
each one being already a little closer to the vertex than the previous
one. And it is for example \( \infty \) if there are no signals at all or there are
only stationary signals.
If each signal moves with its velocity and upon reaching a vertex propagates to all incident edges except the one it came from (making copies of itself if necessary) and if collisions of signals are ignored, then the set of points an event occurs in at a time in the future is given a name in

**Definition 4.12.** Let \( c \) be a configuration of \( \text{Cnf} \) and let \( t \) be a time of \( T \). The set of vertices that signals in \( c \) reach at time \( t \) (under the assumptions given in the introduction to the present definition) is

\[
M' = \begin{cases} 
\emptyset, & \text{if } t = 0, \\
\bigcup_{m \in M \atop s \in c(m) \atop \text{spd}(s) > 0} R_{\text{vel}(s) \cdot t}(m) \cap \mathcal{V}, & \text{otherwise.
}}
\]

The set of points that signals in \( c \) collide in at time \( t \) is

\[
M'' = \bigcup_{m, m' \in M \atop m \neq m'} \bigcup_{s \in c(m) \atop s' \in c(m')} R_{\text{vel}(s) \cdot t}(m) \cap R_{\text{vel}(s') \cdot t}(m').
\]

The set of points that signals in \( c \) are involved in an event in at time \( t \) is \( M_t(c) = M' \cup M'' \).

**Remark 4.13.** For times \( t \) before and including the time \( t_0(c) \) of the next event, the definition of \( M_t(c) \) is natural. And for other times, it is plausible with the explanation given before the definition and it is used to handle accumulations of events and accumulations of accumulations of events and so on.

As above, if each signal moves with its velocity and upon reaching a vertex propagates to all incident edges except the one it came from (making copies of itself if necessary) and if collisions of signals are ignored, then starting our signal machine in a configuration \( c \) and letting it run for a time \( t \) without handling propagation of signals in vertices at time \( t \) yields a new configuration \( \oplus(t)(c) \) as defined in

**Definition 4.14.**

\[
\oplus : T \to (\text{Cnf} \to \text{Cnf}), \\
0 \mapsto \text{id}_{\text{Cnf}}, \\
t \mapsto \left[c \mapsto [m \mapsto \{s \in \text{Sgnl} \mid \exists m' \in M \exists s' \in c(m') : (m, \text{dir}(s)) \in R_{\text{vel}(s') \cdot t}(m'), \text{knd}(s) = \text{knd}(s'), \text{and} \ dt(s) = \text{dt}(s')\}].
\]

**Remark 4.15.** For each time \( t \in T \) and each configuration \( c \in \text{Cnf} \), if there is a signal in \( c \) that reaches a vertex in time \( t \) (or one of its...
duplicates does), then in the configuration $\ominus(t)(c)$ that signal is at the vertex and its velocity is the one it had just before reaching the vertex. The direction of that velocity points away from the edge the signal came from and it does not point to any edge that is incident to the vertex. It is then up to the local transition function to decide what to do with the signal and, if it is not removed, what its direction shall be.

\[\square\]

**Remark 6.38.** The map $\ominus$ is used to determine future configurations before or right until the next event occurs and needs to be handled, and also to make crude predictions of future configurations beyond the next event time by propagating at vertices and ignoring collisions as explained above. These predictions will be used to handle accumulations of events and accumulations of accumulations of events and so on. \[\square\]

Until the first events occur, signals move along edges without colliding. At the time the first events occur, a signal reached a vertex or two signals collided (on a vertex or edge) or multiple such events happened. An event in a vertex is handled by the local transition function $\delta_v$ and on an edge by $\delta_e$. This global behaviour can be described by a map that maps a configuration to the configuration right after the first events occurred and have been handled. This map is given in

**Definition 4.17.**

$$\Delta_0 : \text{Cnf} \rightarrow \text{Cnf},$$

\[
c \mapsto \begin{cases} 
  c, & \text{if } t_0(c) \in \{0, \infty\}, \\
  \{ m \mapsto \begin{cases} 
    c'(m), & \text{if } m \not\in M_{t_0(c)}(c), \\
    \delta_v(\text{dir}(m), c'(m)), & \text{if } m \in M_{t_0(c)}(c) \cap \mathcal{V}, \\
    \delta_e(c'(m)), & \text{if } m \in M_{t_0(c)}(c) \setminus \mathcal{V}, \\
  \end{cases} \}, & \text{otherwise},
\end{cases}
\]

where $c' = \ominus(t_0(c))(c)$.

\[\square\]

**Remark 4.18.** The map $\Delta_0$ maps a configuration to itself if the next event time is 0, which means that event times accumulated at 0, or $\infty$, which means that there is no next event. And it maps a configuration to the configuration that is reached after the first events have been handled, by first using $\ominus$ to determine the configuration in which the events occur and then handling all occurring events with $\delta_v$ and $\delta_e$. \[\square\]

If the next event time is 0, which we call *singularity of order $-1$* (see figure 4.1b), then the machine is sometimes stuck, in the sense that there is no natural way to define what configuration the machine is in at any time in the future, and sometimes the machine can go forward in time, in the sense that there is a natural way to define what
configuration the machine is in at least until some time in the future; the latter case is handled later and largely ignored for now. If the next event time is $\infty$, then the machine does nothing for eternity.

Otherwise, the machine can at least proceed until the next event time and handle the occurring events, which we call singularities of order $0$, and then the next event time may again be $0$, $\infty$, or something in between. It may happen that the next event times are never $0$ or $\infty$ but accumulate at some time in the future, which we call singularity of order $1$ (see figure 4.1a). In that case repeated applications of $\Delta_0$ never reach a configuration at that future time or a time beyond. But we can in a sense take the limit of the sequence of configurations that repeated applications of $\Delta_0$ yield. Yet, it may even happen that singularities of order $1$ accumulate at some time in the future, which we call singularity of order $2$. Again, we can in the same sense as before take the limit of the sequence of configurations at these singularities. It may continue this way ad infinitum. In precise terms this is done in

**Definition 4.19.** The sequence $(t_{j-1}^n)_{n \in \mathbb{N}_0}$, where the $n$ in $t_{j-1}^n$ is an upper index and does not stand for exponentiation, the map $t_j$, and the map $\Delta_j$, for $j \in \mathbb{N}_+$, are defined by mutual induction as follows: The maps $t_0$ and $\Delta_0$ have already been defined and, for each positive integer $j$, let

$$t_{j-1}^n : \operatorname{Cnf} \to \mathbb{T}, \quad c \mapsto \sum_{i=0}^{n-1} t_{j-1}(\Delta_{j-1}^i(c)),$$

(note that $t_{0-1}^0 = 0$), let

$$t_j : \operatorname{Cnf} \to \mathbb{T}, \quad c \mapsto \lim_{n \to \infty} t_{j-1}^n(c),$$

and let

$$\Delta_j : \operatorname{Cnf} \to \operatorname{Cnf},$$

$$c \mapsto \begin{cases} c, & \text{if } t_j(c) \in \{0, \infty\}, \\ \{m \mapsto \begin{cases} c'(m), & \text{if } m \notin M_j(c), \\ \delta_e(\text{dir}(m), c'(m)), & \text{if } m \in M_j(c) \cap \mathfrak{M}, \\ \delta_e(c'(m)), & \text{if } m \in M_j(c) \setminus \mathfrak{M}, \\ \end{cases} \end{cases}, \\ \end{cases}$$

where $c' = \liminf_{n \to \infty} (t_j^c - t_{j-1}^n(c))(\Delta_{j-1}^n(c))$, and $M_j(c) = \liminf_{n \to \infty} M_{t_j(c)-t_{j-1}^n(c)}(\Delta_{j-1}^n(c))$, where the first limit inferior is the pointwise limit inferior of sequences of set-valued maps and the second limit inferior is the limit inferior
(a) A singularity of order 1.

(b) A singularity of order \(-1\).

Figure 4.1: Both subfigures depict a space-time diagram of the time evolution of an unspecified signal machine, where space is drawn on the horizontal axis and time on the vertical axis evolving from top to bottom. In figure 4.1a, there are infinitely many signals of various speeds arbitrarily close to 0 and slower than 1 emanating from the left vertex, the fastest signal, which is the one of speed 1, is reflected at the right vertex, and the reflected signal collides with all other signals at shorter and shorter time spans between collisions, resulting in a singularity of order 1 at the last depicted time. In figure 4.1b, there are infinitely many signals of various speeds arbitrarily close to 0 and slower than 1 emanating to the right from the middle vertex, and there is one signal of speed 1 emanating to the left from the middle vertex, this signal is reflected at the left vertex, and at the time the reflected signal reaches the middle vertex is a singularity of order \(-1\) because it collides with all signals that emanated to the right and are arbitrarily close to the vertex.
of sequences of sets. In greater detail, for each sequence \((c_n)_{n \in \mathbb{N}_0}\) of set-valued maps from \(M\) to \(Q\), the pointwise limit inferior of \((c_n)_{n \in \mathbb{N}_0}\) is the map \(c: M \to Q\), \(m \mapsto \liminf_{n \to \infty} c_n(m)\) and is denoted by \(\liminf_{n \to \infty} c_n\); and, for each sequence \((A_n)_{n \in \mathbb{N}_0}\) of subsets of \(M\), the limit inferior of \((A_n)_{n \in \mathbb{N}_0}\) is the subset \(\bigcup_{n \in \mathbb{N}_0} \bigcap_{k \geq n} A_k\) of \(M\) and is denoted by \(\liminf_{n \to \infty} A_n\).

**Remark 4.20.** Let \(c\) be a configuration of Cnf. Then, for each positive integer \(j\), we have \(t_{j-1}^0(c) = 0\) and \(t_{j-1}^1(c) = t_{j-1}(c)\), and the sequence \((t_{j-1}^n(c))_{n \in \mathbb{N}_0}\) in \(\mathcal{T}\) is non-decreasing and hence converges in \(\mathcal{T}\). And, \(\liminf_{n \to \infty} t_{n+1}(c)\) is the map \(\Delta_j: \mathcal{N} \to \mathcal{T}\) that would be reached at the accumulation time if there were no collisions.

**Remark 4.21.** Let the signal machine be in a configuration \(c\) at time 0 and let there be no future configuration whose next event time is 0 or \(\infty\). The latter is the case if and only if the sequences \((t_{j-1}^n(c))_{n \in \mathbb{N}_0}\) for \(j \in \mathbb{N}_+\), and hence also the sequence \((t_j(c))_{j \in \mathbb{N}_0}\), are strictly increasing sequences in \(\mathcal{T}\).

Then, for each positive integer \(j\) and each non-negative integer \(n\), at time \(t_{j-1}^n(c)\) the machine is in configuration \(\Delta_j^n\). And, the time \(t_{j-1}^0(c)\) is the \(n\)-th time an event occurs (singularity of order 0), the time \(t_{j-1}^1(c)\) is the \(n\)-th time an accumulation of events occurs (singularity of order 1), the time \(t_{j-1}^2(c)\) is the \(n\)-th time an accumulation of accumulations of events occurs (singularity of order 2), and so forth.

Moreover, for each non-negative integer \(j\), at time \(t_j(c)\) the machine is in configuration \(\Delta_j(c)\). And, the time \(t_0(c)\) is the next time an event occurs, the time \(t_1(c)\) is the next time an accumulation of events occurs, the time \(t_2(c)\) is the next time an accumulation of accumulations of events occurs, and so forth.

Furthermore, for each positive integer \(j\), the map \(\Delta_j\) maps the configuration \(c\) to the configuration that is reached after an accumulation of singularities of order \(j-1\), which is a singularity of order \(j\). First, it calculates the configurations that accumulate, namely \(\Delta_{j-1}^n(c)\); secondly, for each of these configurations, it uses \(\boxplus\) to determine the configuration that would be reached at the accumulation time if there were no further events, which is a crude prediction of the future that becomes better the greater \(n\) is; thirdly, it calculates the pointwise limit inferior of these configurations, which is essentially the configuration that contains the signals that all but finitely many of the configurations have in common (in particular, if for a point \(m\) the sequence of signals at \(m\) become constant, then the limit at \(m\) is that set of signals); lastly, it handles collisions.

**Remark 4.22.** Let the signal machine be in a configuration \(c\) at time 0 and let there be a future configuration whose next event time is 0. Then, there is a least positive integer \(j\) such that the sequence \((t_j^n(c))_{n \in \mathbb{N}_0}\)
is eventually constant. And, there is a least non-negative integer \( n \) such that \( t^n_{j-1}(c) = t^{n+1}_{j-1}(c) \). The time \( t' = t^n_{j-1}(c) \) is the first time at which the signal machine is in a configuration whose next event time is 0 and this configuration is \( c' = \Delta^n_{j-1}(c) \).

For each non-negative integer \( n' \) such that \( n' \geq n \), we have \( t^{n'}_{j-1}(c) = t' \) and \( \Delta^{n'}_{j-1}(c) = c' \). And, for each positive integer \( j' \) such that \( j' \geq j \), the time \( t_{j'}(c) \) is equal to \( t' \) and the configuration \( \Delta_{j'}(c) \) is equal to \( c' \), and the sequences \( (t_{j'}^n(c))_{n \in \mathbb{N}_0} \) and \( (\Delta_{j'}^n(c))_{n \in \mathbb{N}_0} \) are the constant sequences \( (t')_{n \in \mathbb{N}_0} \) and \( (c')_{n \in \mathbb{N}_0} \). In particular, the limit \( \lim_{j \to \infty} t_{j'}(c) \) is equal to \( t' \) and the limit inferior \( \liminf_{j \to \infty} \Delta_{j'}(c) \) is equal to \( c' \). □

**Remark 4.23.** The limit of sequences of configurations and of sets of points does in general not exist. However, the limit inferior and the limit superior always exist. We decided not to use the limit, to avoid case distinctions that would have to be made. Instead, we decided to use the limit inferior; we could as well have decided to use the limit superior. Which of the two has the desired outcome depends on the specific use case.

For the signal machine that solves the firing squad synchronisation problem that we construct in the next section and the configurations it is initialised with and the configurations it encounters during its time evolution, the encountered limit inferiors and superiors are actually always the same, which means that the limits exist, and hence the choice of limit inferior or superior is irrelevant in that use case. □

If the machine never assumes a configuration in which events accumulate at time 0 and if non-negative singularities of ever higher orders ad infinitum do not accumulate, then the machine can be observed for eternity. Otherwise, it can for now only be observed for all times before \( \lim_{j \to \infty} t_{j'}(c) \), where \( c \) is the initial configuration of the machine. This time is given a name in

**Definition 4.24.** For each configuration \( c \in \text{Cnf} \), the non-negative real number or infinity \( t_{\infty}(c) = \lim_{j \to \infty} t_{j'}(c) \) is called \( \infty \)-existence time of \( c \), and the closed interval \([0, t_{\infty}(c)]\) is called \( \infty \)-existence interval of \( c \). □

Repeated applications of powers of the maps \( \Delta_j \), for decreasing \( j \in \mathbb{N}_0 \), let us jump to and from configurations right after singularities of decreasing orders. The resulting map is given a name in

**Definition 4.25.** For each non-negative integer \( j \), each non-negative integer \( k \), and each finite sequence \((n_i)_{i=j}^k\) of non-negative integers, let

\[
\Delta_{(n_i)_{i=j}^k} = \begin{cases} 
\text{id}_{\text{Cnf}}, & \text{if } j > k, \\
\Delta_j^{n_j} \circ \Delta_{j+1}^{n_{j+1}} \circ \cdots \circ \Delta_k^{n_k}, & \text{otherwise},
\end{cases}
\]

map \( \Delta_{(n_i)_{i=j}^k} \) from \( \text{Cnf} \) to \( \text{Cnf} \).
and let
\[ t_{(n_i)_{i=j}}^k = \sum_{i=j}^k t_i^{n_i} \circ \Delta_{(n_i)_{i=j}}^{k+1}. \]

**Remark 4.26.** Let the signal machine be in a configuration \( c \) at time 0 and let there be no future configuration whose next event time is 0 or \( \infty \). The map \( \Delta_{(n_i)_{i=j}}^{k} \) applied to \( c \), first applies \( \Delta_k^{n_k} \) to jump from \( c \) to the configuration right after the \( n_k \)th time a singularity of order \( k \) occurs, secondly it applies \( \Delta_{k-1}^{n_{k-1}} \) to jump from that configuration, namely \( \Delta_{(n_i)_{i=k}}^{k} (c) \), to the configuration right after the \( n_{k-1} \)th time a singularity of order \( k - 1 \) occurs (counting from the time at which the machine is in configuration \( \Delta_{(n_i)_{i=k}}^{k} (c) \), and so forth until it finally applies \( \Delta_j^{n_j} \) to jump from the configuration \( \Delta_{(n_i)_{i=j}}^{j} (c) \) to the configuration right after the \( n_j \)th time a singularity of order \( j \) occurs (counting from the time at which the machine is in configuration \( \Delta_{(n_i)_{i=j}}^{j} (c) \)), where in the case that \( j = 0 \), a singularity of order 0 is nothing but an event.

The time it takes the machine to get from \( c \) to the configuration \( \Delta_k^{n_k} (c) \) is \( t_k^{n_k} (c) \), the time it takes to get from \( \Delta_k^{n_k} (c) \) to \( \Delta_{k-1}^{n_{k-1}} (\Delta_k^{n_k} (c)) \) is \( t_{k-1}^{n_{k-1}} (\Delta_k^{n_k} (c)) \), and so forth: in total, the time it takes to get from \( c \) to \( \Delta_{(n_i)_{i=j}}^{k} (c) \) is \( t_{(n_i)_{i=j}}^k (c) \).

To compute the configuration the machine is in at the \( \infty \)-existence time of the initial configuration, we can use the maps \( \Delta_j \) for increasing \( j \) to jump from singularities of non-negative lower orders to singularities of ever higher orders, which in the case there are any singularities of order \(-1\) comes to a halt at the first such singularity at the \( \infty \)-existence time and in the other case yields in a sense the limit configuration. And, to compute the configuration at time \( t \) before the \( \infty \)-existence time, we can use one of the maps \( \Delta_{(n_i)_{i=0}}^{k} \) to jump to the configuration right after the last event before time \( t \) and then we can use the map \( \boxplus \) to jump from there to time \( t \). The resulting map describes the time evolution of the signal machine before or at \( \infty \)-existence times and it is given in

**Definition 4.27.** For each time \( t \in \mathbb{T} \), the set of configurations whose \( \infty \)-existence interval contains \( t \) is
\[ \text{Cnf}_t^\infty = \{ c \in \text{Cnf} \mid t \leq t_\infty (c) \}. \]

Let
\[ \boxplus: \mathbb{T} \to \bigcup_{t \in \mathbb{T}} (\text{Cnf}_t^\infty \to \text{Cnf}), \]
\[
Cnf_t^\infty \to Cnf,
\]
\[
t \mapsto \begin{cases} 
\liminf_{j \to \infty} \Delta_j(c), & \text{if } t = t_\infty(c), \\
\exists (t - t_{(n_i)}_{i=0}^k(c))(\Delta_{(n_i)}^k(c)), & \text{otherwise,}
\end{cases}
\]

for the least \( k \in \mathbb{N}_0 \) and the \( (n_i)_{i=0}^k \in \mathbb{N}_0^{k+1} \) with \( t \in [t_{(n_i)}_{i=0}^k(c), (n_0+1,n_2, \ldots, n_k)(c)] \).

Note that the least index \( k \) and the finite sequence \( (n_i)_{i=0}^k \) that occur above are uniquely determined by the time \( t \) and the configuration \( c \).

\[\square\]

**Remark 4.28.** In the second case in the definition of \( \square \), we have \( t \in [t_k^{n_k}(c), t_k^{n_k+1}(c)] \), and \( t - t_k^{n_k}(c) \in [t_k^{n_k-1}(\Delta_k^{n_k}(c)), t_k^{n_k-1+1}(\Delta_k^{n_k}(c))] \), and so forth.

\[\square\]

**Remark 4.29.** For each configuration \( c \in Cnf \), the map \( \square(\_)(c) \) is defined on the closed interval \([0, t_\infty(c)]\).

\[\square\]

**Remark 4.30.** Let \( t \) be a time of \( T \) and let \( c \) be a configuration of \( Cnf_t^\infty \) such that \( t \neq t_\infty(c) \). If events do not accumulate for the initial configuration \( c \), then \( \square(t)(c) = \exists (t - t_0^{n_0}(c))(\Delta_0^{n_0}(c)) \), for the \( n_0 \in \mathbb{N}_0 \) with \( t \in [t_0^{n_0}(c), t_0^{n_0+1}(c)] \); in words, we apply \( \Delta_0 \) repeatedly, jumping from event to event, until we reach the configuration \( \Delta_0^{n_0}(c) \) at time \( t_0^{n_0}(c) \) with the property that the next event (if there even is one) occurs after \( t \), at which point we use \( \square \) to move signals along edges for the remaining time \( t - t_0^{n_0}(c) \).

If events do accumulate for \( c \) but singularities of order 1 do not accumulate, then we apply \( \Delta_1 \) repeatedly, jumping from singularity to singularity, until we reach the configuration \( \Delta_1^{n_1}(c) \) at time \( t_1^{n_1}(c) \) with the property that the next singularity (if there even is one) occurs after \( t \), at which point we apply \( \Delta_0 \) repeatedly, jumping from event to event, until we reach the configuration \( \Delta_0^{n_0}(\Delta_1^{n_1}(c)) \) at time \( t_0^{n_0}(c) + t_1^{n_1}(c) \) with the property that the next event (if there even is one) occurs after \( t \), at which point we use \( \square \) to move signals along edges for the remaining time \( t - t_0^{n_0}(c) - t_1^{n_1}(c) \).

And so forth.

\[\square\]

**Remark 4.31.** Let the signal machine be in a configuration \( c \) at time 0. If there is a singularity of order \(-1\) in the future, then, according to remark 4.22, the time \( t_\infty(c) \) is the first time at which the signal machine is at such a singularity and the corresponding configuration is \( \square(t_\infty(c))(c) \). Otherwise, the time \( t_\infty(c) \), which may be the improper time \( \infty \), is the time just after all singularities and the corresponding configuration is \( \square(t_\infty(c))(c) \). In either case, in what is to come, for simplicity, if \( t_\infty(c) \) is finite, then we talk as if there is a singularity of order \(-1\) at time \( t_\infty(c) \).

\[\square\]
At the time of a singularity of order 1, there are infinitely many events that occur just before that time and arbitrarily close to it, and the problem is to define the configuration at the time of the singularity. At the time of a singularity of order $-1$, there are infinitely many events that occur just after that time and arbitrarily close to it, and the problem is to define the configurations at all times after the singularity.

Analogous problems exist for accumulations of singularities of order 1 or $-1$, and accumulations of accumulations of singularities of order 1 or $-1$, and so forth. For singularities of positive orders these problems have been solved above but not for singularities of order $-1$ and its accumulations.

At a singularity of order $-1$ at time 0, to compute the configuration at a small enough time $t$, first, we make crude predictions of the future with $\boxplus$ by jumping past the singularity to future times $\varepsilon$ ignoring events, secondly, we extrapolate these predictions to the time $t$ with $\boxminus$ by letting the machine evolve them until the time $t$, and, lastly, we take the limit inferior of these predictions as $\varepsilon$ tends to 0. This does not work for all singularities of order $-1$ regardless of how small we choose $t$ (for example if at each point in time of a time span after and including 0 an event occurs), but it does work for the singularities of order $-1$ that occur in our quasi-solution of the firing squad synchronisation problem.

If the $\infty$-existence time of the current configuration is $\infty$, then the machine can be observed for eternity. Otherwise, it can be observed until the $\infty$-existence time using $\boxminus$ and from there at least until the least time until which all the crude predictions mentioned above can be observed; this time span is named in

**Definition 4.32.**

\[ t_{-1}: \text{Cnf} \rightarrow \mathbb{T}, \]
\[ c \mapsto \begin{cases} \infty, & \text{if } t_\infty(c) = \infty, \\ \inf_{\varepsilon \in \mathbb{R}_{>0}} (\varepsilon + t_\infty(c)) + t_\infty(c_\varepsilon), & \text{otherwise}, \end{cases} \]
where $c_\varepsilon = \boxplus(\varepsilon)(\boxminus(t_\infty(c))(c))$. \[\square\]

How the machine evolves until and beyond $t_\infty$ and at most until $t_{-1}$ is given in

**Definition 4.33.** For each time $t \in \mathbb{T}$, let
\[ \text{Cnf}_{t-1} = \{ c \in \text{Cnf} \mid t \leq t_{-1}(c) \} \]
and let

\[ \boxplus: \mathbb{T} \rightarrow \bigcup_{t \in \mathbb{T}} (\text{Cnf}_{t-1} \rightarrow \text{Cnf}), \]
where the limit inferior is the pointwise limit inferior of set-valued maps, in greater detail, for each family \( \{c_\varepsilon\}_{\varepsilon \in \mathbb{R}_{>0}} \) of set-valued maps from \( M \) to \( Q \), the pointwise limit inferior \( \liminf_{\varepsilon \downarrow 0} c_\varepsilon \) is the map \( c : M \to Q, m \mapsto \bigcup_{r \in \mathbb{R}_{>0}} \bigcap_{s \geq r} c_1 s(m) \).

**Remark 4.34.** For each configuration \( c \in \text{Cnf} \), we have \( t_\infty(c) = t_{-1}(c) \) for each time \( t \in T \), we have \( \text{Cnf}^t \subseteq \text{Cnf}^{-1} \); and for each configuration \( c \in \text{Cnf} \) and each positive real number \( \varepsilon \), we have \( t_{-1}(c) - (\varepsilon + t_\infty(c)) \leq t_\infty(c_\varepsilon) \).

**Remark 4.35.** Let the signal machine be in a configuration \( c \) at time 0. If \( t_\infty(\square(t_\infty(c)))(c) = 0 \), then there is a singularity of order \(-1\) at time \( t_\infty(c) \). Otherwise, there is not. In the latter case, as one would hope, \( t_{-1}(c) = t_\infty(\square(t_\infty(c)))(c) \), and, for each time \( t \in [t_\infty(c), t_{-1}(c)] \), we have \( \square(t)(c) = \square(t - t_\infty(c))(\square(t_\infty(c)))(c) \).

Unlike for a singularity of a non-negative order, for a singularity of order \(-1\) it is not possible to jump to the time right after the singularity as there is no such time. However, we may jump over the singularity at time \( t_\infty \) to the time \( t_{-1} \). This is done by the map

**Definition 4.36.**

\[ \Delta_{-1} : \text{Cnf} \to \text{Cnf}, \]

\[
c \mapsto \begin{cases} c, & \text{if } t_\infty(c) = \infty, \\ \square(t_{-1}(c))(c), & \text{otherwise}. \end{cases} \]

Like for singularities of non-negative orders, such of order \(-1\) may accumulate, which is a singularity of order \(-2\), such of order \(-2\) may accumulate, which is a singularity of order \(-3\), and so forth. The map to jump over singularities of order \(-1\) has already been given and the maps to jump over singularities of smaller orders are introduced in

**Definition 4.37.** The sequence \( (t_{-(j-1)})_{n \in \mathbb{N}_0} \), where the \( n \) in \( t_{-(j-1)} \) is an upper index and does not stand for exponentiation, the map \( t_{-j} \), and the map \( \Delta_{-j} \), for \( j \in \mathbb{Z}_{\geq 2} \), are defined by mutual induction as follows: The maps \( t_{-1} \) and \( \Delta_{-1} \) have already been defined and, for each integer \( j \) such that \( j \geq 2 \), let

\[
\begin{align*}
\left( t_{-(j-1)} : \text{Cnf} \to T, \quad c \mapsto \sum_{i=0}^{n-1} t_{-(j-1)}(\Delta_{-(j-1)}^i(c)) \right)_{n \in \mathbb{N}_0}
\end{align*}
\]
(note that \( t^0_{-(j-1)} = 0 \)), let

\[
\begin{align*}
t_{-j}: \text{Cnf} &\rightarrow \overline{T}, \\
c &\mapsto \lim_{n \to \infty} t^n_{-(j-1)}(c),
\end{align*}
\]

and let

\[
\begin{align*}
\Delta_{-j}: \text{Cnf} &\rightarrow \text{Cnf}, \\
c &\mapsto \begin{cases} 
  c, & \text{if } t_{-j}(c) = \infty, \\
  \liminf_{n \to \infty} (t_{-j}(c) - t^n_{-(j-1)}(c))(\Delta^n_{-(j-1)}(c)), & \text{otherwise}.
\end{cases}
\end{align*}
\]

The machine can be observed for all times before \( \lim_{j \to \infty} t_{-j}(c) \), where \( c \) is the initial configuration of the machine. This time is given a name in Definition 4.38.

**Definition 4.38.** For each configuration \( c \in \text{Cnf} \), the non-negative real number or infinity \( t_{-\infty}(c) = \lim_{j \to \infty} t_{-j}(c) \) is called \((-\infty)\)-existence time of \( c \), and the closed interval \([0, t_{-\infty}(c)]\) is called \((-\infty)\)-existence interval of \( c \).

Like for singularities of non-negative orders, repeated applications of the maps \( \Delta_{-j} \), for decreasing \( j \in \mathbb{N}_+ \), let us jump over singularities of decreasing negative orders down to order \(-1\). The resulting map is given a name in Definition 4.39.

**Definition 4.39.** For each positive integer \( j \), each non-negative integer \( k \), and each finite sequence \((n_i)_{i=-j}^{-k}\) of non-negative integers that is indexed from \(-j\) down to \(-k\), let

\[
\Delta_{(n_i)_{i=-j}^{-k}} = \begin{cases} 
  \text{id}_{\text{Cnf}}, & \text{if } -j < -k, \\
  \Delta_{-j} \circ \Delta_{-j-1} \circ \cdots \circ \Delta_{-k}, & \text{otherwise},
\end{cases}
\]

and let

\[
\begin{align*}
t_{(n_i)_{i=-j}^{-k}}: \text{Cnf} &\rightarrow \overline{T}, \\
&= \sum_{i=-j}^{-k} t^n_{i} \circ \Delta_{(n_i)_{i=-i-1}^{-i-1}}.
\end{align*}
\]

Like for singularities of non-negative orders, to compute the configuration the machine is in at the \((-\infty)\)-existence time of the initial configuration, we can use the maps \( \Delta_{-j} \) for increasing \( j \) to jump from singularities of negative lower orders to singularities of ever higher orders ad infinitum. And, to compute the configuration at time \( t \) before the \((-\infty)\)-existence time, we can use one of the maps \( \Delta_{(n_i)_{i=-j}^{-k}} \) to jump over all the singularities before time \( t \) such that the next jump over a singularity of order \(-1\) would be beyond time \( t \) and then we can use...
the map to jump from there to time \( t \) whereby we may cross a singularity of order \(-1\). The resulting map describes the time evolution of the signal machine before or at \(( -\infty )\)-existence times, which for the purposes of this treatise is the complete time evolution, and this map is given in

**Definition 4.40.** For each time \( t \in T \), the set of configurations whose \(( -\infty )\)-existence interval contains \( t \) is

\[
\operatorname{Cnf}^{-\infty}_{t} = \{ c \in \operatorname{Cnf} \mid t \leq t_{-\infty}(c) \}.
\]

The map

\[
\Box : T \rightarrow \bigcup_{t \in T} (\operatorname{Cnf}^{-\infty}_{t} \rightarrow \operatorname{Cnf}),
\]

\[
t \mapsto \begin{cases} \lim_{j \to \infty} \Delta^{-j}(c), & \text{if } t = t_{-\infty}(c), \\ \Box (t - t_{(n_{i})_{i=1}^{k}}(c))(\Delta_{(n_{i})_{i=1}^{k}}^{-k}(c)), & \text{otherwise}, \end{cases}
\]

for the least \( k \in \mathbb{N}_{+} \) and the \((n_{i})_{i=1}^{k} \in \mathbb{N}_{0}^{k} \)

with \( t \in \left[t_{(n_{i})_{i=1}^{k}}(c), t_{(n_{i=1}+1, n_{i=2}, n_{i=3}, \ldots, n_{k})}(c)\right] \)

is called global transition function. \( \square \)

**Remark 4.41.** Jérôme Olivier Durand-Lose introduces and studies another notion of signal machines in his paper ‘The signal point of view: from cellular automata to signal machines’[Dur2]. The notable differences are the following: While our machines are defined over continuum graphs, his machines are defined over the real number line; while our machines may have infinitely many different kinds of signals, his machines may only have signals of a finite number of kinds (which he calls meta-signals); while in a configuration of our machines there may exist infinitely many signals (even at the same point), in configurations of his machines there may only exist finitely many signals; while the time evolution of our machines can be observed beyond singularities of any order, the time evolution of his machines already stops before singularities of order 1, that is, before accumulations of collisions (as there are no vertices, other events do not exist for his machines). \( \square \)

5 **Firing Squad Signal Machines**

In this section, let \( \mathcal{G} = (V, E, \varepsilon) \) be a non-trivial, finite, and connected undirected multigraph, let \( \omega \) be an edge weighting of \( \mathcal{G} \), let \( M \) be a continuum representation of \( \mathcal{G} \), and let \( g \) be a vertex of \( M \), which we call general. When we say path, we either mean path in \( \mathcal{G} \) or path in \( M \); when we say longest path, we either mean maximum-weight path in \( \mathcal{G} \) or longest path in \( M \); and, when we say path, we often mean non-empty

\[
\text{general vertex } g
\]
direction-preserving path from vertex to vertex; in all cases it should be clear from the context what is meant.

From a broad perspective, the signal machine we construct in this section performs the following tasks: It cuts the graph such that the graph turns into a virtual tree; it starts synchronisation of edges as soon as possible and freezes it as late as possible; it determines the midpoints of all non-empty direction-preserving paths from vertices to vertices; it determines which midpoints are the ones of the longest paths; starting from the midpoints of the longest paths, it traverses midpoints of shorter and shorter paths and upon reaching midpoints of edges, it thaws synchronisation of the respective edges; all edges finish synchronisation at the same time with the creation of fire signals that lie dense in the graph (see figures 7.3 to 7.7). A more detailed account is given in

**Remark 5.1.**  

A. Turn the graph into a tree: Initiate signals of speed 1 spread from the general throughout the graph without making U-turns and vanish at leaves. When they collide on an edge but not in one of its ends, the edge is cut in two at the point of collision by two stationary leaf signals (also called *virtual leaves*), one for each of the two new ends. When they collide in a vertex coming from all incident edges, all edges are cut off by leaf signals; and when the do not come from all incident edges, all but one of the edges the signals come from are cut off. In the latter case, the initiate signal that comes from the edge that is not cut off spreads onto the edges from which no initiate signals came.

In this way all cycles are eventually broken up and the graph is turned into a virtual tree. Because the virtual leaves are created as soon as possible and because they are treated just like normal leaves, we may and will assume in the description of the other tasks that the graph is a tree.

B. Start and freeze synchronisation of edges (see figure 5.2): When initiate signals reach a vertex, for each incident edge, synchronisation of the edge is started from the vertex immediately and freezing of this synchronisation is started from the midpoint of the edge after \( \frac{3}{2} \) times the edge’s length time units by sending freeze signals of speed 1 to both ends of the edge and finishes after twice times the edge’s length time units.

Each edge is synchronised by recursively dividing it into two parts, one having two-third its length and the other having one-third its length. This division procedure becomes finer and finer, in other words, divisions accumulate, the closer the time evolution gets to the time 2 times the edge’s length after of the edge’s synchronisation started.
Figure 5.1: Both subfigures depict a space-time diagram of the time evolution of the signal machine that we construct in section 5. The diagram in figure 5.1a illustrates how, beginning from one end of an edge, the midpoint of the edge is found, which works analogously for paths instead of edges, and the diagram in figure 5.1b illustrates how, beginning from the inner vertex of a path consisting of two edges, the midpoint of the path is found, which works analogously for longer paths; above the right space-time diagram is a scaled-down depiction of the two-edged path. Vertices are thick and solid lines; find-midpoint signals of speed 1 are thick and dotted lines; reflected find-midpoint signals of speed 1 are densely dotted lines; slowed-down find-midpoint signals of speed 1/3 are solid lines; stationary midpoint signals are thick and solid lines.
Figure 5.2: Both subfigures depict a space-time diagram of the time evolution of the signal machine that we construct in section 5. The diagram in figure 5.2a illustrates how, beginning from one end of an edge, the edge is synchronised by recursively dividing it into two parts, where one part is two-thirds the superpart’s length and the other part is one-third the superpart’s length, and creating fire signals when reflected divide signals reach the boundaries they originated at, which happens for all at the same time and these fire signals lie dense in the edge. And, the diagram in figure 5.2b illustrates how, beginning from the inner vertex of a path consisting of two edges, the path is synchronised by synchronising both edges, freezing the synchronisation of individual edges as late as possible and thawing it as early as possible such that both edges finish at the same time. Note that in both subfigures, at each boundary, there is a singularity of order 1 at the last depicted time, and in figure 5.2b, there are additional singularities of order 1 at time twice the longer edge’s length and at time twice the shorter edge’s length, and of order −1 at various points of the freeze and thaw signals. Only the most relevant signals are depicted and these only for the most relevant time spans. Initiate signals, divide signals of type 0, and find-midpoint signals, all of speed 1, are solid lines; reflected divide signals and reflected find-midpoint signals, both of speed 1, are solid lines; divide signals of type \( n \in \mathbb{N}_+ \), which have speed \( (2/3)^n/(2 - (2/3)^n) \), are densely dotted lines; slowed-down find-midpoint signals of speed 1/3 are solid lines; stationary boundary and midpoint signals are thick and solid lines; freeze signals of speed 1 are dashed lines; thaw signals of speed 1 that do not thaw synchronisation of the edge they are on are thick and loosely dotted lines, and the other thaw signals of speed 1 are dashed lines.
Figure 5.3: Both subfigures depict the space-time diagram of the time evolution of the signal machine that we construct in section 5 for the scaled-down depicted tree with two edges, beginning with the initial configuration for the firing squad synchronisation problem — which is the configuration in which initiate signals, find-midpoint signals, and slowed-down find-midpoint signals emanate from the general onto all incident edges — and ending with the final configuration of the problem — which is the configuration in which the fire signals lie dense in the multigraph —, where in figure 5.3a the general is the vertex that is incident to both edges and in figure 5.3b it is the leaf of the longer edge. Only the most relevant signals are depicted and these only for the most relevant time spans. Initiate signals and find-midpoint signals, which in the depicted cases always travel alongside each other, are thick and dotted lines; reflected find-midpoint signals are densely dotted lines; slowed-down find-midpoint signals are solid lines; midpoint signals are thick and solid lines; freeze signals are dashed lines; thaw signals that do not thaw synchronisation of the edge they are on are thick and loosely dotted lines, and the other thaw signals are dashed lines. The synchronisation of individual edges, before it is frozen and after it is thawed, is schematically represented by hatch patterns.
Figure 5.4: The figure depicts the space-time diagram of the time evolution of the signal machine that we construct in section 5 for the tree with three edges depicted on the right, beginning with the initial configuration for the firing squad synchronisation problem — which is the configuration in which initiate signals, find-midpoint signals, and slowed-down find-midpoint signals emanate from the general onto all incident edges — and ending with the final configuration of the problem — which is the configuration in which the fire signals lie dense in the multigraph —, where the general is the second vertex from the left. Only the most relevant signals are depicted and these only for the most relevant time spans. Initiate signals and find-midpoint signals are thick and dotted lines; reflected find-midpoint signals are densely dotted lines; slowed-down find-midpoint signals are solid lines; midpoint signals are thick and solid lines; freeze signals are dashed lines; thaw signals that do not thaw synchronisation of the edge they are on are thick and loosely dotted lines, and the other thaw signals are dashed lines. When a midpoint is found and when the corresponding midpoint signal collides with a matching thaw signal, the path it is the midpoint of is represented by the colours of the path’s edges in a little disk or rounded rectangle. The synchronisation of individual edges, before it is frozen and after it is thawed, is schematically represented by hatch patterns.
Figure 5.5: Both subfigures depict the space-time diagram of the time evolution of the signal machine that we construct in section 5 for the scaled-down depicted tree with three edges, beginning with the initial configuration for the firing squad synchronisation problem — which is the configuration in which initiate signals, find-midpoint signals, and slowed-down find-midpoint signals emanate from the general onto all incident edges — and ending with the final configuration of the problem — which is the configuration in which the fire signals lie dense in the multigraph. In figure 5.5a the general is the one that is incident to all edges; and in figure 5.5b it is the leaf of the shortest edge. Only the most relevant signals are depicted and these only for the most relevant time spans. Each edge has a colour, leaves are coloured according to the edge they are incident to, and signals and hatch patterns are coloured according to the edge they are on. The shortest edge is depicted twice, once coloured red and overlaying the green edge, and once coloured violet and overlaying the blue edge; this makes it easier to follow signals that travel from or onto the shortest edge. Initiate signals and find-midpoint signals, which in the depicted cases always travel alongside each other, are thick and dotted lines; reflected find-midpoint signals are densely dotted lines; slowed-down find-midpoint signals are solid lines; midpoint signals are thick and solid lines; freeze signals are dashed lines; thaw signals that do not thaw synchronisation of the edge they are on are thick and loosely dotted lines, and the other thaw signals are dashed lines. When a midpoint is found and when the corresponding midpoint signal collides with a matching thaw signal, the path it is the midpoint of is represented by the colours of the path's edges in a little disk. The synchronisation of individual edges, before it is frozen and after it is thawed, is schematically represented by hatch patterns.
The division of one part is performed by sending divide signals of speed \( (2/3)^n/(2 - (2/3)^n) \), for \( n \in \mathbb{N}_0 \), from one boundary onto the part, reflecting the divide signal of speed 1 at the other boundary, and, upon collision of the reflected divide signal with a divide signal, letting the reflected divide signal move on, removing the involved divide signal, creating a stationary boundary signal and sending divide signals of the above speeds onto the newly created part the reflected divide signal comes from.

C. Determine the midpoints of all direction-preserving paths from vertices to vertices (see figure 5.1): The general and initiate signals that reach vertices send find-midpoint signals of speed 1 and slowed-down find-midpoint signals of speed 1/3 in all directions to determine the midpoints of all paths that contain the general and the reached vertex respectively. These signals spread throughout the graph without making U-turns memorising the paths they take and (fast) find-midpoint signals are additionally reflected.

Reflected find-midpoint signals of speed 1 take the paths back they took before being reflected memorising the path to the vertex they were reflected at and the remaining path they have to take. Upon finishing their path at the vertex they originated at, slowed-down find-midpoint signals of speed 1/3 are sent onto all edges except the one the reflected signal comes from, they memorise the path from the origin vertex to the reflection vertex of the reflected find-midpoint signal, and they spread throughout the graph without making U-turns memorizing the paths they take and vanish at leaves.

When a reflected find-midpoint signal collides with a slowed-down find-midpoint signal that originated at the same vertex, the point of collision is the midpoint of the concatenation of the paths the two signals took after being reflected, that is, the path from the vertex the reflected find-midpoint signal was reflected at over the point of collision over the vertex both signals originated at to the vertex the reflected find-midpoint signal that spawned the slowed-down find-midpoint signal was reflected at. Each midpoint is designated by a stationary midpoint signal that memorises the path it is the midpoint of along with its position on the path.

When a reflected find-midpoint signal collides with a reflected find-midpoint signal that originated at the same vertex and the point of collision is the origin vertex itself, both reflection vertices have the same distance to the origin vertex and this vertex is the midpoint of the concatenation of the paths the two signals took after being reflected.

To get a clearer picture of how midpoints are determined, let us focus on only a few signals and let us ignore boundary cases: When an initiate signal reaches a vertex, one find-midpoint signal is sent
along one incident edge, and another find-midpoint signal is sent along another incident edge. Both signals travel along edges and upon reaching a vertex they are either reflected and travel back or they take one of the incident edges that leads them further away. At the latest, they are reflected upon reaching a leaf. One of the reflected find-midpoint signals returns first to the vertex it originated at, is slowed down there, and the slowed down signal travels towards the other (reflected) find-midpoint signal. At some time in the future, the slowed-down find-midpoint signal collides with the other, now reflected, find-midpoint signal and the point of collision is the midpoint of the concatenation of the paths the two signals took after being reflected.

D. Determine the midpoints of the longest direction-preserving paths (see figures 5.3 and 5.4): The midpoints of the longest paths are eventually found. But this is not sufficient, they also need to be recognised as such. To that end, each reflected find-midpoint signal and each slowed-down find-midpoint signal carries a boolean that indicates whether the path it took from the vertex it was reflected at may be the subpath of a longest path that has the same source (or target), and whether the signal would be the first one to find the longest path’s midpoint (recall that a slowed-down find-midpoint signal was either spawned by the general vertex or an initiate signal that reached a vertex, in which case we can think of this vertex as the one the slowed-down find-midpoint signal was reflected at; or it was originally a reflected find-midpoint signal and knows where that signal was reflected at). Let us call signals that carry the boolean yes marked and the others not.

At each vertex that is not a leaf, a stationary count signal memorises the directions from which marked reflected find-midpoint signals that originated at the vertex or from which marked slowed-down find-midpoint signals with any origin have already returned. Because longest paths always end at leaves, for each leaf, find-midpoint signals are marked when they are the first ones to be reflected at the leaf and not otherwise. When a marked signal reaches a vertex, it stays marked, if, including itself, from all but one direction have marked signals already returned (which is the case if and only if it is the last signal to return from its direction and the penultimate signal to return among such signals from either direction), and it is unmarked, otherwise.

This means that a marked reflected find-midpoint signal is unmarked, if there are at least two other signals that started out at the same time at the same vertex but take longer to return (because they have a longer way; the combined paths that two of the other signals take may be a longest path), and it stays marked, if all other signals except for one that all started out at
the same time at the same vertex returned earlier (because they had a shorter way).

And it means that a marked slowed-down find-midpoint signal is unmarked, if it is unclear which of the incident edges belong to longest paths, and it stays marked, otherwise, which is the case if from precisely one direction no slowest reflected find-midpoint signal that originated at the vertex has returned yet. This is not only pessimistic, meaning that we do not falsely consider midpoints of paths that are not among the longest as such, but also correct, meaning that we still find the midpoints of all longest paths in time (see section 6.2).

When a marked reflected find-midpoint signal reaches its origin and is the penultimate such signal to do so, it turns into a marked slowed-down find-midpoint signal that travels in the one direction from which no marked signal has returned yet and the point at which this signal collides with the one signal that has not yet returned will be the midpoint of a longest path, if both signals are still marked at the time of collision.

E. Traverse midpoints, thaw synchronisation of edges, and fire (see figures 5.4 and 5.5): The midpoints of the longest paths are found at the same time, at which, from each such midpoint, two thaw signals of speed 1 are sent that travel along the midpoint’s path towards both of its ends. When a thaw signal collides with the midpoint of a path such that one of the two subpaths from the midpoint to either end of the path coincides with the remaining path the thaw signal travels along, an additional thaw signal is created that travels along the other subpath. On its way from the midpoint of the last edge of the path a thaw signal travels on to the end of the path, the thaw signal thaws all frozen signals it collides with. All thaw signals reach the ends of their paths at the same time, which is also the time all edges finish synchronisation with the creation of stationary fire signals that lie dense in the graph.

We introduce a typographic convention in

Definition 5.2. Each word of letters of the Latin alphabet that is written in typewriter font shall denote the word itself and shall for example not be the name of a variable.

We introduce a boolean algebra in

\[
\text{booleans} \
B = \{\text{no, yes}\}
\]

Definition 5.3. Let \( B = \{\text{no, yes}\} \). Each element \( b \in B \) is called boolean. The map

\[
\neg: B \to B,
\]

38
\[
\begin{align*}
\text{no} &\mapsto \text{yes}, \\
\text{yes} &\mapsto \text{no},
\end{align*}
\]
is called negation. The map
\[
\land : \mathcal{B} \times \mathcal{B} \to \mathcal{B},
\]
\[
(b, b') \mapsto \begin{cases} 
\text{no}, & \text{if } \text{no} \in \{b, b'\}, \\
\text{yes}, & \text{otherwise}, 
\end{cases}
\]
is called conjunction.

We introduce finite lists of directions in

**Definition 7.6.** Let \(\text{Dir}^*\) be the set \(\{w : [1 : n] \to \text{Dir} \mid n \in \mathbb{N}_0\}\). Each element \(w \in \text{Dir}^*\) is called word over Dir; for each word \(w \in \text{Dir}^*\), the non-negative integer \(|w| = |\text{dom}(w)|\) is called length of \(w\); the word \(\lambda : \emptyset \to \text{Dir}\) is called empty; the map
\[
\bullet : \text{Dir}^* \times \text{Dir}^* \to \text{Dir}^*,
\]
\[
(w, w') \mapsto \begin{bmatrix} [1 : |w| + |w'|] \to \text{Dir}, \\
& \begin{cases} w(i), & \text{if } i \leq |w|, \\
w'(i - |w|), & \text{otherwise}, \end{cases} \end{bmatrix}
\]
is called concatenation.

**Remark 7.5.** The empty word is the only word of length 0 and it is the neutral element of \(\bullet\). □

The signal machine we construct in this section has infinitely many kinds of signals, which are explained in remark 5.8, and given names, speeds, and data sets in

**Definition 5.6.** Let
\[
\text{Knd} = \{I, L, C, M, U, \bar{U}, V, F, T\} \cup \left( \bigcup_{n \in \mathbb{N}_0} \{D_n, FD_n\} \right) \cup \{\bar{D}, B, X, FX\},
\]
let
\[
\text{spd} : \text{Knd} \to \mathbb{R}_{\geq 0},
\]
\[
I \mapsto 1, \\
L \mapsto 0, \\
C \mapsto 0, \\
M \mapsto 0, \\
U \mapsto 1, \\
\bar{U} \mapsto 1, \\
V \mapsto \frac{1}{3},
\]

39
\[ F \mapsto 1, \]
\[ T \mapsto 1, \]
\[ D_n \mapsto \left(\frac{2}{3}\right)^n, \text{ for } n \in \mathbb{N}_0, \]
\[ FD_n \mapsto 0, \text{ for } n \in \mathbb{N}_+, \]
\[ \vec{D} \mapsto 1, \]
\[ B \mapsto 0, \]
\[ X \mapsto 0, \]
\[ FX \mapsto 0, \]

and let

\[ \text{set } \text{None} = \{0\}, \]
\[ \text{family } \{Dt_k\}_{k \in \text{Knd}}, \]
\[ \begin{array}{l}
Dt_I = \text{None}, \\
Dt_L = \text{Dir}, \\
Dt_C = \mathcal{P}(\text{Dir}), \\
Dt_M = \{\{w, w'\} \subseteq \text{Dir}^* | w \neq w'\}, \\
Dt_0 = \text{Dir}^*, \\
Dt_q = \text{Dir}^* \times \text{Dir}^* \times \text{B}, \\
Dt_v = \text{Dir}^* \times \text{Dir}^* \times \text{B}, \\
Dt_F = \text{None}, \\
Dt_T = \text{Dir}^* \times \text{B}, \\
Dt_{D_n} = \text{None}, \text{ for } n \in \mathbb{N}_0, \\
Dt_{FD_n} = \text{Dir}, \text{ for } n \in \mathbb{N}_+, \\
Dt_B = \text{None}, \\
Dt_X = \text{None}, \\
Dt_{FX} = \text{None}. 
\end{array} \]

\[ \square \]

The kinds together with their speeds and data sets determine the possible signals, which are recalled and given abbreviations in

**Definition 5.7.** The set of signals is

\[ Sgnl = \{(k, d, u) | k \in \text{Knd}, \ (\text{spd}(k), d) \in \text{Arr}, \text{ and } u \in Dt_k\}. \]

Let

\[ \begin{array}{l}
\text{set } Sgnl, \\
\text{abbreviations of signals like } I_d, \\
\text{and } FX.
\end{array} \]

\[ \begin{array}{l}
I_d = (I, d, 0), \text{ for } d \in \text{Dir}, \\
L_d = (L, \text{vry}, d), \text{ for } d \in \text{Dir}, \\
C_D = (C, \text{vry}, D), \text{ for } D \in Dt_C, \\
M_{\{w, w'\}} = (M, \text{vry}, \{w, w'\}), \text{ for } \{w, w'\} \in Dt_M, \\
\end{array} \]
\[ u_{w_o,d} = (U, d, w_o), \text{ for } w_o \in D_u, \]
\[ \bar{u}_{w_o,d,w_r} = (\bar{U}, d, (w_o, w_r, b)), \text{ for } d \in \text{Dir} \text{ and } (w_o, w_r, b) \in D_\bar{u}, \]
\[ v_{d,w_o,w_r} = (V, d, (w_o, w_r, b)), \text{ for } d \in \text{Dir} \text{ and } (w_o, w_r, b) \in D_v, \]
\[ F_d = (F, d, 0), \text{ for } d \in \text{Dir}, \]
\[ T_{d,w} = (T, d, (w, b)), \text{ for } d \in \text{Dir} \text{ and } (w, b) \in D_t, \]
\[ D_{n,d} = (D_n, d, 0), \text{ for } n \in \mathbb{N}_0 \text{ and } d \in \text{Dir}, \]
\[ FD_{n,d} = (FD_n, \text{vry}, d), \text{ for } n \in \mathbb{N}_+ \text{ and } d \in \text{Dir}, \]
\[ \bar{B}_d = (\bar{B}, \text{vry}, 0), \]
\[ X = (X, \text{vry}, 0), \]
\[ FX = (FX, \text{vry}, 0). \]

What signals of various kinds do when they reach a vertex or collide with one another, what the data they carry means, and what we call them is given a glimpse at in Remark 7.

**Remark 5.8.**  
A. Each signal of kind \( I \) has speed 1; at each vertex it reaches it spreads in all directions that lead away from where it comes from, it initiates synchronisation of all incident edges except the one it comes from by sending divide signals onto them, it initiates the search for midpoints of paths that contain the vertex by sending (slowed-down) find-midpoint signals in all directions, and it initiates one component of the search for the longest paths of the graph by marking slowed-down find-midpoint signals if the vertex is a leaf; it carries no data; and it is called **initiate signal**. The very first initiate signals spread from the general in all directions.

**initiate signal**

B. Each signal of kind \( L \) is stationary, designates a virtual leaf, carries the direction that leads onto the edge that is incident to the virtual leaf, and is called **leaf signal**. Such signals are created when initiate signals collide in a vertex (or on an edge), which means that there is a cycle in the graph, and this cycle is broken up by virtually terminating the involved edge(s) with leaf signals. Each leaf signal is treated like a leaf in the following way: When signals collide with each other and with leaf signals, for each involved leaf signal, the collision of the signals that move in the opposite direction than the one the leaf signal carries is handled as if those signals collided in a leaf. Because leaf signals are created at points at the same time or before any other signal reaches them, the graph looks like a tree for all other signals.

**leaf signal**

C. Each signal of kind \( C \) is stationary; is positioned at a vertex that is not a leaf; memorises the directions from which find-midpoint signals that originated at the vertex, were reflected, and may
be on longest paths and would be the first ones to find their midpoints have already returned, in other words, it memorises the directions from which the slowest find-midpoint signals that originated at the vertex and travelled alongside initiate signals before they were reflected at a leaf have already returned; and is called \textit{count signal}. When an initiate signal reaches a vertex that is not a leaf, a count signal is created.

Note that, for the data set of count signals, instead of the infinite set $\mathcal{P}(\text{Dir})$, we could have used the finite set $\mathcal{P}(\{1, 2, \ldots, k\})$, where $k$ is the maximum degree of the graph or the upper bound of the maximum degrees of the graphs to be considered and the numbers represent directions that lead away from vertices. The first choice of $k$ would make the signal machine depend on the graph, which is unconventional, whereas the latter would not and would also fit to the fact that solutions of firing mob synchronisation problems are usually considered for graphs whose maximum degrees are uniformly bounded by a constant.

\begin{itemize}
  \item[D.] Each signal of kind $\mathcal{M}$ is stationary, designates the midpoint of a path, carries the directions that lead from its position to both ends of the path, and is called \textit{midpoint signal}. Such a signal is created when a reflected find-midpoint signal collides with a slowed-down find-midpoint signal that originated at the same vertex, or when two reflected find-midpoint signals that originated at the same vertex collide with each other, which only happens at the origin vertex itself. See figure 5.1.
  \item[E.] Each signal of kind $\mathcal{U}$ has speed 1; at each vertex it reaches it spreads in all directions that lead away from where it comes from (in the sense that, in each such direction, a signal of its kind is sent) and it is also reflected (in the sense that a reflected find-midpoint signal is sent in the direction from where it comes from), it carries the directions that lead from its position to the vertex the signal originated at, and it is called \textit{find-midpoint signal}. When an initiate signal reaches a vertex, for each incident edge, a find-midpoint signal whose origin is the vertex is created that travels onto the edge.
  \item[F.] Each signal of kind $\mathcal{U}$ has speed 1; is the reflection of a find-midpoint signal at a vertex, travels back along the path this signal took before it was reflected and slows down when it reaches the vertex the find-midpoint signal originated at; carries the directions that lead from its position to the vertex the find-midpoint signal originated at, the directions that lead from its position to the vertex the find-midpoint signal was reflected at, and a boolean that indicates whether the path described by its position and both directions, which leads from the reflection vertex to the origin vertex, may be the subpath of a longest path that
has the same source (or target), and the boolean also indicates whether the signal would be the first one to find the longest path’s midpoint; and is called \textit{reflected find-midpoint signal} and, if the boolean it carries is \textit{yes}, it is called \textit{marked}.

As has already been pointed at, when a find-midpoint signal reaches a vertex, a reflected find-midpoint signal is created that travels onto the edge the find-midpoint signal comes from. If the vertex is a leaf and the find-midpoint signal is one of the first signals to reach it, which is precisely the case if the signal reaches the leaf together with an initiate signal, then its reflection is marked, and otherwise, not. The reasons are that both ends of a longest path are leaves and that a find-midpoint signal that is not among the first signals to reach one end of a longest path would not find its midpoint after another one has already found it.

When a marked reflected find-midpoint signal reaches a vertex that is not a leaf, the count signal at the vertex memorises the direction the marked signal comes from, and the signal stays marked, if the memory of the count signal contains each but one direction that leads away from the vertex, and it is unmarked, otherwise. Why is that? Each vertex that is not the general is reached precisely once by an initiate signal, at which point find-midpoint signals are sent in all directions; for each direction, the marked reflected find-midpoint signal to return from that direction is memorised, which is the slowest one or, in other words, the last one or the one that had the longest way (note that although only one find-midpoint signal is sent in a direction, multiple reflected find-midpoint signals may return from that direction); the penultimate marked reflected find-midpoint signal to return may come from one edge of a longest path that runs through the vertex and hence it stays marked (note that the other edge of the longest path that is incident to the vertex would be the one from which the marked signal has not yet returned); the signals that return before the penultimate one are too fast to be on a longest path and the last signal to return has already collided with the slowed-down penultimate signal that returned before it (if they do not return at the same time) and hence they are unmarked.

G. Each signal of kind $\mathcal{V}$ has speed $1/3$; is the slow-down of a reflected find-midpoint signal at the vertex the find-midpoint signal originated at; at each vertex it reaches it spreads in all directions that lead away from where it comes from; it carries the directions that lead from its position to the vertex the find-midpoint signal originated at, the directions that lead from the origin vertex to the vertex the find-midpoint signal was reflected at, and a boolean that indicates whether the path described by its position and both directions, which leads from the reflection vertex over the origin
vertex to its position, may be the subpath of a longest path that has the same source (or target), and the boolean also indicates whether the signal would be the first one to find the longest path’s midpoint; and it is called \textit{slowed-down find-midpoint signal} and, if the boolean it carries is \textbf{yes}, it is called \textit{marked}.

As has already been pointed at, when a reflected find-midpoint signal reaches the vertex the find-midpoint signal originated at, for each incident edge except the one the reflected find-midpoint signal comes from, a slowed-down find-midpoint signal is created that travels onto the edge. Additionally, when an initiate signal reaches a vertex, for each incident edge, a slowed-down find-midpoint signal is created that travels onto the edge.

The latter case is in the following senses the boundary or limiting case of the former: Imagine that the vertex the initiate signal reaches is in fact two vertices that are infinitesimally close; then a find-midpoint signal is created at one vertex, this signal immediately reaches the infinitesimally close other vertex, there it is reflected, the reflected find-midpoint signal immediately reaches the infinitesimally close other vertex, and there it is slowed down. Or, analogously, imagine the limit of the cascade of the creation of a find-midpoint signal, its reflection, and slow-down for shorter and shorter distances between the vertex the find-midpoint signal originates at and the one it is reflected at; then in the limit the find-midpoint signal and its reflection vanish and only the slowed-down find-midpoint signal remains.

When a marked slowed-down find-midpoint signal reaches a vertex that is not a leaf, the count signal at the vertex memorises the direction the marked signal comes from, and the signal stays marked, if the memory of the count signal contains each but one direction that leads away from the vertex, and it is unmarked, otherwise. Why is that? The slowed-down signal reaches the vertex at the same time and from the same direction as the slowest reflected find-midpoint signal from that direction that originated at the vertex. The latter signal is however not marked, because it did not travel alongside initiate signals before it was reflected at a leaf (the reason is that if it had travelled alongside initiate signals, then it would have been reflected at the same time as the find-midpoint signal whose reflection turned into the marked slowed-down find-midpoint signal and hence, because the paths from the reflection leaves to the vertex they reach together have the same lengths, the find-midpoint signal that reaches it slowed down would have taken longer, and therefore the signals would not reach the vertex at the same time).

Therefore, the memory of the count signal contains each but one direction if and only if from each but one direction the slowest
reflected find-midpoint signals that originated at the vertex have already returned. If this is the case, then the incident edge belonging to that direction may be the edge of a longest path that runs through the vertex and the slowed-down find-midpoint signal may be the one to collide with the not yet returned signal somewhere on or beyond the edge precisely at the midpoint of the longest path. If from more than one direction signals are overdue, then the paths running through each pair of these directions are longer than the paths running through any of these directions and the direction the slowed-down find-midpoint signal comes from. And, if all signals have already returned, then they have already collided with the slowed-down signal and found the midpoints of the longest paths whose determination involves the slowed-down signal if there are any.

H. Each signal of kind \( F \) has speed 1, is created at the midpoint of an edge, moves towards one end of the edge, and freezes synchronisation of the edge, carries no data, and is called \textit{freeze signal}. When an initiate signal reaches a vertex, for each incident edge, a find-midpoint signal and a slowed-down find-midpoint signal are created that travel onto the edge, the former is reflected at the other end of the edge and collides with the latter at the midpoint of the edge, at which point two freeze signals are created that travel to both ends of the edge. See figure 5.3.

I. Each signal of kind \( T \) has speed 1, is created at the midpoint of a path, travels along the path towards one of end of the path, creates a new signal of its kind when it collides with the midpoint signal that designates the midpoint of a path such that one of the two subpaths from the midpoint to either end of the path (a half-path) coincides with the path it takes itself and the new signal travels along the other half-path, and thaws synchronisation of an edge if it collided with or was created at the midpoint signal that designates the midpoint of the last edge of the path it takes, carries the directions of the path it takes and a boolean that indicates whether it thaws synchronisation of the edge it is on or not, and is called \textit{thaw signal}.

The first thaw signals are created simultaneously at the midpoints of longest paths. For each such midpoint, two thaw signals are created, one that travels along the path to one end of the path and the other that travels to the other end of the path. When a thaw signal collides with the midpoint of a path whose one half-path coincides with the remaining path the thaw signal travels along, an additional thaw signal is created that travels along the other half-path. On its way from the midpoint of the last edge of the path a thaw signal travels on to the end of the path, the thaw signal thaws all frozen signals it collides with.
In this way, starting at the midpoints of longest paths, thaw signals traverse the midpoints of shorter and shorter paths, reach the ends of their paths at the same time, and thaw synchronisation of edges, which finishes at the same time. See figures 5.3 to 5.5.

J. Each signal of kind $D_0$ has speed 1, moves from one boundary (which may be one end of an edge or a boundary signal) to the next boundary (which may be the other end of the edge or a boundary signal) and is reflected there, carries no data, and is called divide signal of type 0. When an initiate signal reaches a vertex, for each incident edge, a divide signal of type 0 is created that travels onto the edge. And, when a divide signal of any type collides with a reflected divide signal, a divide signal of type 0 is created that travels in the same direction as the (non-reflected) divide signal.

K. Each signal of kind $D_n$, for $n \in \mathbb{N}_+$, has speed $(2/3)^n/(2 - (2/3)^n)$, moves from one boundary (which may be one end of an edge or a boundary signal) towards the next boundary (which may be the other end of the edge or a boundary signal) but never reaches it and can be frozen, carries no data, and is called divide signal of type $n$. When an initiate signal reaches a vertex, for each incident edge, and for each $n \in \mathbb{N}_+$, a divide signal of type $n$ is created that travels onto the edge. And, when a divide signal of any type collides with a reflected divide signal, for each $n \in \mathbb{N}_+$, a divide signal of type $n$ is created that travels in the same direction as the (non-reflected) divide signal.

Note that although $D_n$, for $n \in \mathbb{N}_+$, are different kinds, events that involve signals of these kinds are handled the same way, in other words, signals of these kinds are not differentiated by the two local transition functions of the signal machine. The only reason they are different kinds is because we need them to have different speeds and by definition all signals of the same kind have the same speed.

L. Each signal of kind $FD_n$, for $n \in \mathbb{N}_+$, has speed 0, is a frozen divide signal of type $n$, carries the direction the non-frozen divide signal had, and is called frozen divide signal of type $n$. When a freeze signal collides with or is created at the same time as a divide signal of type $n \in \mathbb{N}_+$, the divide signal is frozen.

M. Each signal of kind $\ddot{D}$ has speed 1, is the reflection of a divide signal of type 0, creates a boundary signal when it collides with a divide signal of type $n$, creates a fire signal when it reaches the end of the edge it traverses, carries no data, and is called reflected divide signal. When a divide signal of type 0 reaches a boundary (which may be a vertex or a boundary signal), a reflected divide signal is created that travels in the opposite direction.
N. Each signal of kind B is stationary, designates a boundary for the synchronisation of an edge, carries no data, and is called boundary signal. Such signals are created when divide signals collide with reflected divide signals.

On each edge, the interplay of divide signals, reflected divide signals, and boundary signals has the following effect: At first a divide signal of type 1 collides with a reflected divide signal that originated at the same end of the edge. This collision results in the creation of a boundary signal that divides the edge into two parts. The length of the part from the origin vertex to the boundary signal is $2/3$ times the length of the edge and the length of the part from the boundary signal to the other end of the edge is $1/3$ times the length of the edge.

In the same manner as the edge itself, the $1/3$-part is recursively divided further and further. In the $2/3$-part, a signal of type 2 collides with the reflected divide signal from before. This collision results in the creation of a boundary signal that divides the $2/3$-part into two subparts. One has $2/3 \cdot 2/3$ the length of the edge and the other has $1/3 \cdot 2/3$ the length of the edge. The $(1/3 \cdot 2/3)$-part is recursively divided further and further. The $(2/3 \cdot 2/3)$-part is divided into a $((2/3) \cdot 2/3 \cdot 2/3)$-part and a $((1/3 \cdot 2/3) \cdot 2/3)$-part and so forth.

If there were no freeze and thaw signals, after twice the time the edge is long — which is precisely the time it took the divide signal of type 0 to reach the other end of the edge, to be reflected there, and to return to the end it originated at — the boundary signals together are dense on the edge, which means that each point on the edge is arbitrarily close to a boundary signal, and, at this point in time, each boundary signal collides with a reflected divide signal, which results in the creation of fire signals that designate that synchronisation has finished.

However, because the synchronisation of each edge is started at different times and takes different times depending on how far away the edge is from the general and how long the edge is, synchronisation of each edge is frozen at the last possible moment — the freezing starts from the midpoint of the edge $3/2$ times the edge’s length many time units after synchronisation of the edge was initiated — and it is thawed such that all edges finish synchronisation at the same time — the thawing starts from the midpoint of the edge $1/3$ times the edge’s length many time units before the total synchronisation finishes, which is the sum of the radius of the graph and its diameter. Recall that the radius is the longest distance from the general to another vertex and that the diameter is the longest distance between two vertices.
Note that, for each edge, collisions of divide signals with reflected divide signals and with boundary signals accumulate at the times the two freeze signals and the two thaw signals reach the ends of the edge. These accumulations are singularities of order 1. See figure 5.2.

O. Each signal of kind $X$ is stationary, designates that synchronisation has finished and can be frozen, carries no data, and is called fire signal. Such signals are created when reflected divide signals collide with boundary signals or reach vertices.

P. Each signal of kind $FX$ is stationary, is a frozen fire signal, carries no data, and is called frozen fire signal. On each edge, one of the two freeze signals reaches an end of the edge at the same time as the reflected divide signal, at which point a frozen fire signal is created; it is thawed at the same time at which all other fire signals are created, which happens on all edges at the same time and the fire signals lie dense in the multigraph.

In the forthcoming definitions of maps we make extensive use of pattern matching. To make the exposition concise and readable we introduce some pattern matching conventions in

A. In the case that patterns of multiple rules overlap, the rule that occurs first is the one to use. For example, the map $f : \mathbb{Z} \to \mathbb{Z}$, $0 \mapsto 1$, $1 \mapsto 0$, $z \mapsto z$, maps 0 to 1 and 1 to 0 and each other integer to itself.

B. In the case that we do not care to name some part of a matched structure, we write _ instead of a name for the part. For example, the pattern $(E, _, I_d)$ matches each triple whose last component is an initiate signal, gives the first component the name $E$, does not give a name to the second component, and gives the direction of the initiate signal the name $d$. And, the pattern $U_{wo, _}$ matches each find-midpoint signal, gives the directions to the vertex the signal originated at the name $w_o$, but gives no name to the direction of the signal.

C. To express equality of different parts of a matched structure, we give those parts the same name. For example, the pattern $\{\tilde{D}_{-d}, D_{n,d}\}$ matches each set that consists of a reflected divide signal and a divide signal of any type such that the direction of the reflected divide signal is the opposite of the direction of the divide signal, gives the direction of the divide signal the name $d$, and gives its type the name $n$.

D. To name both a structure and its parts, we use a Haskell-like @-notation. For example, the pattern $s@U_{\lambda,d,\lambda}$ matches each reflected find-midpoint signal whose directions to the vertex the
signal originated at and to the vertex the signal was reflected at
are empty, gives the direction of the signal the name $d$, gives the boolean that indicates whether the signal may be on a longest
path and would be the first to find its midpoint the name $b$, and
gives the signal itself the name $s$.

Some of the maps we define below are actually partial maps. We
represent them by (total) maps as specified in

**Definition 5.10.** Let $X$ and $X'$ be two sets, let $Y$ be a subset of $X$,
let $\bot$ be an element that is not in $X'$, which we call bottom, and let $f$
be a map from $X$ to $X' \cup \{ \bot \}$ such that, for each element $x \in X$, we
have $f(x) = \bot$ if and only if $x \notin Y$. The map $f$ represents a partial
map whose domain of definition is $Y$, whose domain is $X$, and whose
codomain is $X'$.

In the following, for maps like $f$, we do not explicitly specify the
domain $Y$ of definition (it is the set $X \setminus f^{-1}(\bot)$) and we implicitly
assume that $\bot$ does not occur in the codomain $X'$ of the represented
partial map.

To define the local transition functions, we begin with definitions for
special cases and use those to gradually arrive at definitions for the
general case. For trees and without freezing and thawing, the map $\delta^{\text{tree}}_{v,1}$
handles the event that precisely one signal reaches a vertex, and the
maps $\delta^{\text{tree}}_{v,2}$ and $\delta^{\text{tree}}_{e,2}$ handle the event that precisely two signals collide
in a vertex and an edge respectively (see definition 5.11). For trees, the
maps $\delta^{\text{tree}}_{v}$ and $\delta^{\text{tree}}_{e}$ handle events involving arbitrarily many signals
by considering unordered pairs of signals and applying $\delta^{\text{tree}}_{v,1}$, $\delta^{\text{tree}}_{v,2}$, and
$\delta^{\text{tree}}_{e,2}$, and by also freezing and thawing signals if needed (see defini-
tion 5.12). For virtual trees, which means that edges of the graph have
been virtually cut by leaf signals to remove circles, the maps $\delta^{\text{virtualTree}}_{v}$
and $\delta^{\text{virtualTree}}_{e}$ handle events by partitioning signals at virtual cuts into
those belonging to one or the other leaf and applying $\delta^{\text{tree}}_{v,1}$, $\delta^{\text{tree}}_{v,2}$, and
$\delta^{\text{tree}}_{e,2}$ (see definition 5.13). For general graphs, the maps $\delta_{v}$ and $\delta_{e}$ handle events
by virtually cutting the graph, which eventually creates a virtual tree,
and applying $\delta^{\text{virtualTree}}_{v}$ and $\delta^{\text{virtualTree}}_{e}$ (see definition 5.14).

Most of the forthcoming definitions and parts of them are annotated
with intuitive explanations of what they mean. For example, after each
rule that handles a specific kind of event, it is explained what kind of
event in the time evolution of the signal machine is handled, how it is
handled, and sometimes why.

How events for trees and without freezing and thawing, with one or
two signals involved are handled is given in
**Definition 5.11.** The following map tells whether two words of directions are both empty:

\[ \text{areEmpty}: \text{Dir}^* \times \text{Dir}^* \to \mathbb{B}, \]

\[
(w, w') \mapsto \begin{cases} 
\text{yes}, & \text{if } |w| = 0 \text{ and } |w'| = 0, \\
\text{no}, & \text{otherwise}.
\end{cases}
\]

For trees and without freezing and thawing, the case that precisely two signals collide on an edge but not in one of its ends is handled by the following map, which maps colliding unordered pairs of signals for which a collision rule is specified to the resulting signals and all other sets of signals to \( \bot \):

\[ \delta^{\text{tree}}_{e,2}: \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}) \cup \{\bot\}, \]

\[
\{D_{0,d}, B\} \mapsto \{\tilde{D}_{-d}, B\},
\]

(If a divide signal of type 0 collides with a boundary signal, then reflect the divide signal.)

\[
\{\tilde{D}_{-d}, D_{n,d'}\} \mapsto \{B\} \cup \{D_{n',d'} \mid n' \in \mathbb{N}_0\},
\]

(If a reflected divide signal collides with a divide signal of any type, then create a boundary signal and send divide signals of all types in the direction of the original divide signal.)

\[
\{s@\tilde{U}, s'@\tilde{V}, s''\} \mapsto \begin{cases} 
\{s, s', M_{\{(-d) \cdot w, d \cdot w', b\}}\}, & \text{if } b = \text{no} \text{ or } b' = \text{no}, \\
\emptyset, & \text{otherwise},
\end{cases}
\]

(If a reflected find-midpoint signal collides with a slowed-down find-midpoint signal that originated at the same vertex, then the point of collision is the midpoint of the shortest path in the virtual tree from the vertex the reflected find-midpoint signal was reflected at to the vertex the signal that spawned the slowed-down find-midpoint signal was reflected at and, if this midpoint is not the midpoint of a longest path, then let the signals move on and designate the point by a midpoint signal, and otherwise, send thaw signals along that longest path to both its ends.)

\[
\{T_{d,w}^0, M_{\{d \cdot w, d' \cdot w'\}}\} \mapsto \{T_{\text{areEmpty}(w,w')}^\text{areEmpty}(w,w') \},
\]

50
(If a thaw signal collides with a midpoint signal that designates the midpoint of a path whose one half-path coincides with the path the thaw signal is going to take, then send an additional thaw signal along the other half-path and, if this path is just a directed edge, then make the thaw signals thaw synchronisation of the edge.)

_ $\mapsto \bot$.

(If none of the above happened, then indicate that by returning bottom.)

A signal that reaches a vertex is in a leaf if and only if there is precisely one direction that leads away from the vertex. And a signal that reaches a vertex is the penultimate one to do so if and only if the number of signals that have already returned including the signal itself is one less than the number of directions that lead away from the vertex. The two maps that express this in an abstract way using booleans are

\[
\text{inLeaf}: \mathcal{P}(\text{Dir}) \to \mathbb{B},
\]

\[
E \mapsto \begin{cases} 
\text{no}, & \text{if } |E| \neq 1, \\
\text{yes}, & \text{otherwise},
\end{cases}
\]

and

\[
\text{penultimate}: \mathcal{P}(\text{Dir}) \times \mathbb{N}_0 \to \mathbb{B},
\]

\[
(E, n) \mapsto \begin{cases} 
\text{no}, & \text{if } |E| - 1 \neq n, \\
\text{yes}, & \text{otherwise}.
\end{cases}
\]

For trees and without freezing and thawing, the case that precisely one signal reaches a vertex is handled by the following map, which maps quadruples — consisting of first, the set of directions that lead away from the vertex; secondly, the number of the directions from which the slowest reflected find-midpoint signals that originated at the vertex have already returned or have just arrived; thirdly, a boolean that indicates whether the signal is among the first ones to reach the vertex; and lastly, the signal that reaches the vertex — to the resulting signals:

\[
\delta_{v,1}^{\text{tree}}: \mathcal{P}(\text{Dir}) \times \mathbb{N}_0 \times \mathbb{B} \times \text{Sgnl} \to \mathcal{P}(\text{Sgnl}),
\]

\[
(_{-}, _{-}, _, D_0, d) \mapsto \{\vec{D} - d\},
\]

(If a divide signal of type 0 reaches a vertex, then reflect it.)

\[
(_{-}, _{-}, _, \vec{D}, d) \mapsto \{X\},
\]

(If a reflected divide signal reaches a vertex, then create a fire signal.)

\[
(E, _{-}, _, _{-}, I_d) \mapsto \left( \bigcup_{e \in E \setminus \{-d\}} \{I_e\} \right)
\]

\[
\cup \left( \bigcup_{e \in E \setminus \{-d\}} \{D_{n,e} \mid n \in \mathbb{N}_0\} \right)
\]

\[
\cup \left( \bigcup_{e \in E} \{U_{\lambda,e}, V_{e,\lambda,\lambda}^{\text{inLeaf}(E)}\} \right),
\]

\[51\]
(If an initiate signal reaches a vertex, then send initiate signals onto all incident edges except the one the original initiate signal comes from, send divide signals of all types onto all incident edges except the one the original initiate signal comes from, and send find-midpoint and slowed-down find-midpoint signals onto all incident edges to find all midpoints of paths that contain the vertex, where, in the case that the vertex is a leaf, the slowed-down find-midpoint signal is marked, where the mark means that it may be on a longest path and would be the first to find its midpoint.)

\[
E, \_, b, U_{w_0,d} \mapsto \{\vec{U}^{\text{inLeaf}(E)}_{w_0,-d,\lambda}\} \\
\cup \left( \bigcup_{e \in E \setminus \{-d\}} \{U_{(-d)\bullet w_0,e}\}\right),
\]

(If a find-midpoint signal reaches a leaf, then reflect it and, if it is one of the first signals to reach the leaf, then also mark it as a signal that may be on a longest path and would be the first to find its midpoint. And, if a find-midpoint signal reaches a vertex that is not a leaf, then reflect it and send find-midpoint signals onto all incident edges except the one the original signal comes from.)

\[
E, n, \_, \vec{U}^b_{\lambda,d,w_0} \mapsto \bigcup_{e \in E \setminus \{-d\}} \{V^b_{\lambda,d,w_0,e}\},
\]

(If a reflected find-midpoint signal reaches the vertex it originated at, then send slowed-down find-midpoint signals onto all incident edges except the one the original signal comes from and, if the original signal is marked and is the penultimate marked signal that originated at and has returned to the vertex, then also mark the slowed-down signals as signals that may be on a longest path and would be the first to find their midpoints.)

\[
E, n, \_, \vec{V}^b_{\lambda,d,w_0} \mapsto \{V^b_{\lambda,d,w_0,e}\},
\]

(If a reflected find-midpoint signal reaches a vertex, then it takes the way back it took before it was reflected and, if it is not one of the penultimate marked signals that reaches the vertex, then it is unmarked.)

\[
E, n, \_, V^b_{\lambda,-d,w_0} \mapsto \bigcup_{e \in E \setminus \{-d\}} \{V^b_{\lambda,-d,w_0,e}\},
\]

(If a slowed-down find-midpoint signal reaches a vertex, then send slowed-down find-midpoint signals onto all incident edges except the one the original signal comes from, and mark these signals, if the original signal is marked — in which case it arrives at the same time and from the same direction as the slowest reflected find-midpoint signal that originated at the vertex, which however is not marked because it arrived too late at the leaf it was reflected at — and from exactly one direction the slowest find-midpoint signal that originated at the vertex has not returned yet.)

\[
\_, \_, \_, F_d \mapsto \emptyset,
\]
(If a freeze signal reaches the end of the edge it freezes, then it vanishes.)

\[ (-, -, -, \mathbb{T}_{d, \lambda}^{\text{yes}}) \mapsto \emptyset, \]

(If a thaw signal reaches the end of its path, then it vanishes.)

\[ (-, -, -, \mathbb{T}_{d, e \bullet w}^{\text{no}}) \mapsto \{ \mathbb{T}_{e, w}^{\text{no}} \}, \]

(If a thaw signal reaches a vertex, then it takes the direction that makes it stay on its path.)

\[ (E, -, -, (s, d, y)) \mapsto \begin{cases} \{ (s, d, y) \}, & \text{if } \text{spd}(s) = 0, \\ \bigcup_{c \in E \setminus \{-d\}} \{ (s, c, y) \}, & \text{otherwise.} \end{cases} \]

(A stationary signal in a vertex stays there, and if a non-stationary signal reaches a vertex, then copies of it are sent onto each edge except the one the signal comes from.)

For trees and without freezing and thawing, the case that precisely two signals collide in a vertex is handled by the following map, which maps triples — consisting of first, the set of directions that lead away from the vertex; secondly, the number of the directions from which the slowest reflected find-midpoint signals that originated at the vertex have already returned or have just arrived; and lastly, the set of colliding signals that is supposed to consist of precisely two signals — to the resulting signals, if a collision rule is specified, and to \( \perp \), otherwise:

\[ \delta_{v, 2}^{\text{tree}} : \mathcal{P}(\text{Dir}) \times \mathbb{N}_0 \times \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}) \cup \{ \perp \}, \]

map \( \delta_{v, 2}^{\text{tree}} (-, -, \{ \overline{\mathbb{D}}_d, B \}) \mapsto \{ X \}, \]

(If a reflected divide signal collides with a boundary signal, then create a fire signal.)

\[ (E, n, \{ s' \overline{U}_{(-d') \bullet w', d', w'} \} \mapsto \begin{cases} \delta_{v, 1}^{\text{tree}} (E, n, \emptyset, s) \\ \cup \delta_{v, 1}^{\text{tree}} (E, n, \emptyset, s') \\ \cup \{ M_{(-d') \bullet w', (-d') \bullet w'} \}, \end{cases} \]

\[ \begin{cases} \text{if } b = \text{no} \text{ or } b' = \text{no}, \\ \{ \mathbb{T}_{-d, \text{no}, w}, \mathbb{T}_{-d', \text{no}, w'} \}, \end{cases} \]

otherwise,

(If a reflected find-midpoint signal collides with a slowed-down find-midpoint signal that originated at the same vertex, then the vertex of collision is the midpoint of the shortest path in the virtual tree from the vertex the reflected find-midpoint signal was reflected at to the vertex the signal that spawned the slowed-down find-midpoint signal was reflected at and, if this midpoint is not the midpoint of a longest path, then treat the original signals as if they reached the vertex alone and designate the point by a midpoint signal, and otherwise, send thaw signals along that longest path to both its ends.)

\[ (E, n, \{ s' \overline{U}_{d, w}, s' \overline{U}_{d', w'} \} \mapsto \]

53
\( \delta_{v,1}^{\text{tree}}(E, n, \text{no}, s) \) if \( b = \text{no} \) or \( b' = \text{no} \) or \( n \neq |E| \),
\( \cup \delta_{v,1}^{\text{tree}}(E, n, \text{no}, s') \) otherwise,
\( \cup \{ M_{((d') \bullet w'), (d'') \bullet w''}) \} \)
\( \{ T^{\text{no}}_{-d, w}, T^{\text{no}}_{-d', w'} \} \),

(If two reflected find-midpoint signals that originated at the same vertex collide with each other, then the vertex of collision is the vertex the signals originated at and it is the midpoint of the shortest path in the virtual tree between the vertices the signals were reflected at and, if this midpoint is not the midpoint of a longest path, then treat the original signals as if they reached the vertex alone and designate the point by a midpoint signal, and otherwise, send thaw signals along that longest path to both its ends.)

\( (\ldots, \ldots, \{ T^{\text{no}}_{d, w}, M_{(w \bullet (d' \bullet w'), d'' \bullet w''}) \}) \mapsto \{ T^{\text{no}}_{d', w'}, T^{\text{no}}_{d'', w''} \}, \)

(If a thaw signal collides with a midpoint signal that designates the midpoint of a path whose one half-path coincides with the path the thaw signal is going to take, then send an additional thaw signal along the other half-path.)

\( \mapsto \bot \).

(If none of the above happened, then indicate that by returning bottom.)

How events for trees are handled is given in

**Definition 5.12.** The maps

\[ \xi: \text{Sgnl} \to \text{Sgnl}, \quad D_{n,d} \mapsto \text{FD}_{n,d}, \quad X \mapsto \text{FX}, \quad s \mapsto s, \]

\[ \chi: \text{Sgnl} \to \text{Sgnl}, \quad \text{FD}_{n,d} \mapsto D_{n,d}, \quad \text{FX} \mapsto X, \quad s \mapsto s, \]

freeze and thaw signals that can be frozen and thawed respectively.

The map

\[ \nu: \mathcal{P}(\text{Sgnl}) \times \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}), \]

\[ (S, S') \mapsto \begin{cases} \xi(S'), & \text{if } \exists F_d \in S \cup S' \text{ and } T^{\text{yes}}_{d,\lambda} \in S \cup S', \\ \chi(S'), & \text{if } \exists T^{\text{yes}}_{d,\lambda} \in S \cup S' \text{ and } \exists F_d \in S \cup S', \\ S', & \text{otherwise}, \end{cases} \]

takes a set of old signals and a set of new signals and freezes the new signals, if the old or new signals contain a freeze signal but not a thaw.
signal that thaws the synchronisation of an edge; thaws the new signals, if the old or new signals contain a thaw signal that thaws the synchronisation of an edge but not a freeze signal; and does nothing, otherwise.

The maps

$$\zeta_2 : \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\mathcal{P}(\text{Sgnl})),$$

$$S \mapsto \{\{s, s'\} \subseteq S \mid s \neq s' \text{ and } \delta_{e,2}^{\text{tree}}(\{s, s'\}) \neq \bot\},$$

and

$$\eta_2 : \mathcal{P}(\text{Dir}) \times \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\mathcal{P}(\text{Sgnl})),$$

$$(D, S) \mapsto \{\{s, s'\} \subseteq S \mid s \neq s' \text{ and } \delta_{e,2}^{\text{tree}}(D, x, \{s, s'\}) \neq \bot\},$$

both take a set of signals and return the set of unordered pairs of distinct signals from the given set for which a collision rule is specified in $\delta_{e,2}^{\text{tree}}$ and $\delta_{v,2}^{\text{tree}}$ respectively.

The map

$$\delta_{e}^{\text{tree}} : \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}),$$

$$S \mapsto \nu(S, \left( S \smallsetminus \bigcup_{P \in \zeta_2(S)} P \right) \cup \left( \bigcup_{P \in \zeta_2(S)} \delta_{e,2}^{\text{tree}}(P) \right)),$$

handles collisions of signals on edges by leaving signals for which no pairwise collision rule with any other signal is specified in $\delta_{e,2}^{\text{tree}}$ as is, by applying $\delta_{e,2}^{\text{tree}}$ to each unordered pair of distinct signals for which a collision rule is specified, and by applying $\nu$ to freeze or thaw signals if there are or were any freeze or thaw signals.

The map

$$\kappa : \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Dir}),$$

$$S \mapsto ( \bigcup_{c_D \in S} D ) \cup \{d \in \text{Dir} \mid \exists w_r \in \text{Dir}^r : \overrightarrow{\text{res}}_{\lambda, d, w_r} \in S\} \cup \{d \in \text{Dir} \mid \exists w_o \in \text{Dir}^r \exists w_r \in \text{Dir}^r : \overrightarrow{\text{res}}_{d, w_o, w_r} \in S\}$$

takes a set of signals that are at a vertex and returns the directions from which the slowest reflected find-midpoint signals that originated at the vertex have already returned or have just arrived. The directions from which signals have already returned is memorised by a count signal, and the other directions are the ones from which marked reflected find-midpoint signals that originated at the vertex or marked slowed-down find-midpoint signals with any origin have just arrived (the slowest ones of the latter kind always arrive at the same time and from the same direction as the slowest but unmarked reflected find-midpoint signal that originated at the vertex arrive from that direction). Note that although we take the union of the memories of all count signals, there
is actually gonna be no such signal in leaves (in which case the union is $\emptyset$) and precisely one such signal in each vertex that is not a leaf (in which case the union is the memory stored by this signal).

The map

$$\kappa: \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{B},$$

$$S \mapsto \begin{cases} \text{yes,} & \text{if } \exists d \in \text{Dir} : I_d \in S, \\ \text{no,} & \text{otherwise,} \end{cases}$$

tells whether a set of signals contains any initiate signals or not. It is used by $\delta_v^{\text{tree}}$ to tell whether a find-midpoint signal that reaches a leaf is among the first ones to do so, which is the case if and only if the signal travels alongside an initiate signal.

The map

$$\delta_v^{\text{tree}}: \mathcal{P}(\text{Dir}) \times \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{P}(\text{Sgnl}),$$

$$(D, S) \mapsto \nu(S, \begin{cases} \emptyset, & \text{if } \text{inLeaf}(D) = \text{yes,} \\ \{c_{\kappa(S)}\}, & \text{otherwise,} \end{cases} \cup \delta_v^{\text{tree}}(D, |\kappa(S)|, \kappa(S), S' \cup \bigcup_{P \in \eta_2(D, S')} P) \cup \bigcup_{P \in \eta_2(D, S')} \delta_v^{\text{tree}}(D, |\kappa(S)|, P)),$$

where $S' = S \setminus \{c_D \in S \mid D \subseteq \text{Dir}\}$.

handles events in vertices by updating the memory of the directions from which the slowest reflected find-midpoint signals that originated at the vertex have already returned or have just arrived, by applying $\delta_v^{\text{tree}}$ to non-count signals for which no pairwise collision rule with any other non-count signal is specified in $\delta_v^{\text{tree}}$, by applying $\delta_v^{\text{tree}}$ to each unordered pair of distinct non-count signals for which a collision rule is specified, and by applying $\nu$ to freeze or thaw signals if there are or were any freeze or thaw signals.

How events for virtual trees, where virtual leaves already exist, are handled is given in

**Definition 5.13.** When colliding signals in the graph and the involved directions are partitioned with respect to the virtual tree, some of the components may be degenerated and applying $\delta_e^{\text{tree}}$ and $\delta_v^{\text{tree}}$ to them may have unwanted effects. Such boundary cases are properly handled by the maps

$$\delta_e^{\text{bndryCases}}: \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{P}(\text{Sgnl}),$$

$$S \mapsto \begin{cases} \emptyset, & \text{if } |S| \leq 1, \\ \delta_e^{\text{tree}}(S), & \text{otherwise,} \end{cases}$$

and

$$\delta_v^{\text{bndryCases}}: \mathcal{P}(\text{Dir}) \times \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{P}(\text{Sgnl}),$$

56
\[(D, S) \mapsto \begin{cases} 
\emptyset, & \text{if } D = \emptyset \text{ or } S = \emptyset, \\
\delta_v \text{tree}(D, S), & \text{otherwise.} 
\end{cases}\]

To handle an event involving the signals \(S\) in a virtual tree we do the following: We partition the signals into leaf signals, namely \(S_L\), for each leaf signal \(L_d \in S_L\), the signals coming from the direction \(-d\), namely \(S_d\), and all other signals, namely \(S_o\). And we denote the set of directions that no leaf signal has by \(D'\). Intuitively, \(S_L\) is the set of virtual leaves, \(S_d\) is the set of signals that reach the virtual leaf \(L_d\), and \(D'\) is the set of directions that do not lead away from any virtual leaf. In the configurations we will encounter, in the case of a collision on an edge, there are either no leaf signals, in which case \(S_o = S\), or there are two leaf signals, in which case \(S_o = \emptyset\) (because there are no stationary signals in virtual leaves besides the leaf signals themselves) and hence either the signals simply collide, or one subset of signals reaches one virtual leaf and the other subset reaches the other virtual leaf. And, in the case of a collision in a vertex, signals coming from some edges may reach a virtual leaf and signals coming from other edges may reach the virtual vertex (we call it virtual because some of its original edges have been cut off; the directions onto the ones that have not been cut off are those in the set \(D'\)). Note that some collisions are not collisions in the virtual tree, because the signals came from different directions of virtual cuts. The maps that do what we just explained are

\[
\delta_v \text{virtualTree} : \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}), \\
S \mapsto S_L \cup \delta_v \text{bdryCases}(S_o) \\
\quad \cup \left( \bigcup_{d \in X} \delta_v \text{bdryCases}(\{d\}, S_d) \right),
\]
and

\[
\delta_v \text{virtualTree} : \mathcal{P}(\text{Dir}) \times \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}), \\
(D, S) \mapsto S_L \cup \delta_v \text{bdryCases}(D', S_o) \\
\quad \cup \left( \bigcup_{d \in X} \delta_v \text{bdryCases}(\{d\}, S_d) \right),
\]

where \(X = \{d \in \text{Dir} \mid L_d \in S\}\), the set of directions that lead away from virtual leaves, \(S_L = \{L_d \mid d \in X\}\), the set of virtual leaves, \(\{S_d\}_{d \in X} = \{\{s \in S \mid \text{dir}(s) = -d\}\}_{d \in X}\), for each direction of a virtual leaf, the set of signals that reach the virtual leaf corresponding to the direction moving towards it, \(S_o = S \setminus (S_L \cup (\bigcup_{d \in X} S_d))\), the signals that are not virtual leaves and that do not reach a virtual leaf, and \(D' = D \setminus X\), the set of directions that do not lead away from virtual leaves. □

How events for graphs are handled is given in

**Definition 5.14.** The map

\[\mu : \mathcal{P}(\text{Sgnl}) \to \mathcal{P}(\text{Sgnl}),\]

map \(\mu\)
\[ S \mapsto \{ d \in \text{Dir} \mid I_d \in S \}, \]

takes a set of signals and returns the set of the reverses of the directions
that initiate signals have.

The map

\[ \varphi_e^L : \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{P}(\text{Sgnl}), \]

\[ S \mapsto \begin{cases} 
\emptyset, & \text{if } |\mu(S)| \leq 1, \\
\{ L_d \mid d \in \mu(S) \}, & \text{otherwise,} 
\end{cases} \]

takes a set of signals and returns the empty set, if there is at most one
initiate signal, and the set that consists of a virtual leaf for each initiate
signal, otherwise.

The map

\[ \varphi_v^L : \mathcal{P}(\text{Dir}) \times \mathcal{P}(\text{Sgnl}) \rightarrow \mathcal{P}(\text{Sgnl}), \]

\[ (D, S) \mapsto \begin{cases} 
\emptyset, & \text{if } |\mu(S)| \leq 1, \\
\{ L_d \mid d \in D \}, & \text{if } |\mu(S)| \geq 2 \text{ and } \mu(S) = D, \\
\{ L_d \mid d \in D \setminus \{ d_x \} \}, & \text{if } |\mu(S)| \geq 2 \text{ and } \mu(S) \neq D, 
\end{cases} \text{ for some } d_x \in D \setminus \mu(S), \]

takes a set of directions and a set of signals and returns the empty set,
if there is at most one initiate signal, or the set that consists of a virtual
leaf for each initiate signal, if there are at least two initiate signals and
initiate signals reached the vertex from all incident edges, or the set
that consists of a virtual leaf for each initiate signal but one, otherwise.
Note that right now the choice of \( d_x \) is non-deterministic; however, if
the finite set of directions carried a total order, then we could determin-
istically choose for example the smallest direction; or if continuum rep-
resentations of graphs were embedded in high-dimensional Euclidean
spaces and a Cartesian coordinate system was chosen such that the
occurring directions are unit vectors, then the lexicographic order is a
total order on the set of directions.

The map

\[ \delta_e : \text{dom}(\delta_e) \rightarrow \mathcal{P}(\text{Sgnl}), \]

\[ S \mapsto \delta_{\text{virtualTree}}^e(S \cup \varphi_e^L(S)) \]

handles collisions on edges by creating two virtual leaves, if two ini-
tiate signals collide, which cuts the edge virtually, and then applying
\( \delta_{\text{virtualTree}}^e \) to the maybe new set of signals.

The map

\[ \delta_v : \text{dom}(\delta_v) \rightarrow \mathcal{P}(\text{Sgnl}), \]

\[ (D, S) \mapsto \delta_{\text{virtualTree}}^v(D, S \cup \varphi_v^L(D, S)) \]
handles events in vertices by creating a virtual leaf for each incident edge, if the vertex is not a leaf and initiate signals reached the vertex from all incident edges, or by creating a virtual leaf for each incident edge from which an initiate signal arrived except for one such edge, otherwise, and then applying $\delta_v^{\text{virtualTree}}$ to the maybe new set of signals.

Intuitively, initiate signals are used to turn the graph into a virtual tree by cutting edges at points where such signals collide. These cuts create virtual leaves which are represented by leaf signals. More precisely: When two initiate signals collide on an edge, it is cut by two leaf signals, one for each of the two directions. And when at least two initiate signals collide in a vertex, there are two cases: If initiate signals arrive from all directions, then each incident edge is cut by a leaf signal; otherwise, the incident edges from which initiate signals arrive are cut except for one such edge — the initiate signal from this excluded edge will spread to all edges that have not been cut.

**Main Theorem 5.15.** The signal machine $S = (\text{Knd}, \text{spd}, \{\text{Dt}_k\}_{k \in \text{Knd}}, (\delta_e, \delta_v))$ is a time-optimal quasi-solution of the firing mob synchronisation problem over continuum representations of weighted, non-trivial, finite, and connected undirected multigraphs in the following sense: For each representation $M$ of such a graph, each vertex $g \in M$, for the time $t = r + d$, where $r = \sup_{m \in M} d(g, m)$ is the radius of $M$ with respect to $g$ and $d = \sup_{m, m' \in M} d(m, m')$ is the diameter of $M$, for the instantiation of $S$ for $M$, for the configuration $c \in \text{Cnf}$ such that $c(g) = \bigcup_{d \in \text{dir}(g)} \{I_d\} \cup \{D_{n, d} \mid n \in \mathbb{N}_0\} \cup \{U_{\lambda, d}, V_{\text{inLeaf}(\text{dir}(g))}\}$ and $c\restriction_{M \setminus \{g\}} = \emptyset$, the points in the configuration $\boxdot(t)(c)$ at which a fire signal occurs lies dense in $M$ with respect to the metric $d$, and no fire signals occur in any of the configurations $\boxdot(s)(c)$, for $s \in \mathbb{R}_{\geq 0}$ with $s < t$.

**Proof Sketch.** A proof is sketched in section 6.

**Remark 5.16.** For each positive integer $k$, under the restriction to multigraphs whose maximum degree is bounded by $k$, for each such multigraph, because directions only need to be locally unique (compare remark 3.9), the set $\{1, 2, \ldots, 2k\}$ can be chosen as the set of directions, which makes the data sets of the kinds $L$ and $\text{FD}$ finite and independent of the multigraph, the finite set $P(\{1, 2, \ldots, k\})$ can be chosen as the data set of the kind $C$, and, depending on the diameter $d$ of the multigraph, the finite set of words over $\text{Dir}$ with maximum length $d$ can be chosen as the sets of words over $\text{Dir}$ that occur in the data sets of the kinds $M, U, \bar{U}, V,$ and $T$ — altogether, the data sets of all kinds can be chosen to be finite but some depend on the multigraph.

**Corollary 5.17.** A discretisation of the signal machine $S$ is a time-optimal cellular automaton quasi-solution of the firing mob synchron-
isation problem over non-trivial, finite, and connected undirected multigraphs.

Proof Sketch. Let $G$ be a non-trivial, finite, and connected undirected multigraph. It is made up of paths whose source and target vertices are not of degree 2 and whose other vertices are of degree 2. Each such path together with its inverse can be regarded as an undirected *uber-edge* whose weight is the length of the path and whose ends are the source and target vertices of the path or its inverse. We call vertices that are not of degree 2 *uber-vertices* and vertices that are of degree 2 *under-vertices*.

Signals jump from vertices to vertices along edges. They *collide* when they jump simultaneously from different vertices onto the same vertex or along the same edge but in different directions, or when signals jump onto vertices on which stationary signals reside. Collisions in uber-vertices are handled as collisions in vertices, collisions in under-vertices are handled as collisions on edges (namely on the uber-edges that contain the under-vertices), and collisions in the midst of edges are handled as collisions on edges (namely on the uber-edges that contain the edges). Signals *reach a vertex* when they jump onto an uber-vertex, but not when they jump onto an under-vertex (because the latter just means that they travel along an uber-edge).

When signals collide in a vertex, be it a uber- or under-vertex, the resulting signals are on the vertex. But when signals collide in the midst of an edge, the resulting signals must be distributed onto both its ends depending on their directions. This last case makes it rather cumbersome to write down the local transition functions explicitly. Vertices must be virtually divided into multiple parts: One part that plays the role of the vertex itself and, for each incident edge, an additional part that plays the role of the midpoint of the edge together with the corresponding part of the other end of the edge. And signals must be cleverly distributed onto these parts depending on their direction and how they came into being, and collisions of signals must also be cleverly handled taking the parts the involved signals came from and are on into account.

This discretisation of the signal machine is actually a cellular automaton over the multigraph with appropriate dummy neighbours that are in a dead state (think for example of the multigraph as being embedded in a coloured $S$-Cayley graph with sufficient maximum degree and of the vertices that do not belong the multigraph as being in a dead state).

Remark 7.3: Jacques Mazoyer showed in 1987 that all infinitely many divide signals of type $n$, for $n \in \mathbb{N}_0$, that emanate from the same point can be generated by a cellular automaton with only finitely many states (see [Maz87]). And, as illustrated in remark 5.16, under the restriction to multigraphs whose maximum degrees are uniformly
bounded by a constant, the data sets of all kinds can be chosen to be finite but some depend on the multigraph. Therefore, depending on the multigraph, the discretisation of $S$ is a cellular automaton with a finite number of states. □

**Open Problem 5.19.** Are there time-optimal signal machine and cellular automaton solutions of the firing mob synchronisation problem over non-trivial, finite, and connected undirected multigraphs whose maximum degrees are uniformly bounded by a constant? Or, more specifically, is it possible to adapt the signal machine $S$ (and thereby its discretisation) such that the data sets of all kinds can be chosen to be finite and independent of the multigraph (and thereby making its discretisation have a finite set of states), for example by reducing the number of midpoints that are and need to be determined? □

### 6 Proof Sketch of the Main Theorem

In this section, we sketch a proof of main theorem 5.15. To that end, let $G = (V, E, \varepsilon)$ be a non-trivial, finite, and connected undirected multigraph, let $\omega$ be an edge weighting of $G$, let $M$ be a continuum representation of $G$, identify vertices of $G$ and $M$ and direction-preserving paths from vertices to vertices of $G$ and $M$, and let $g$ be a vertex of $M$, which we call *general*. Furthermore, let $S$ be the signal machine and let $c$ be the initial configuration of the firing mob synchronisation problem from main theorem 5.15, and, whenever we talk about time evolution, we mean the one of $S$ that is in the configuration $c$ at time 0, for example, *at time $t$* either means *in configuration $\square(t)(c)$* or *essentially in configuration $\square(t)(c)$ but before events have been handled*.

To proof main theorem 5.15, we need to ascertain that the signal machine performs the following tasks: First, it cuts the multigraph such that the multigraph turns into a virtual tree and looks like a tree to all other tasks; secondly, it starts synchronisation of edges and freezes it in time; thirdly, it determines the midpoints of all non-empty direction-preserving paths from vertices to vertices in time; fourthly, it determines which midpoints are the ones of the longest paths; fifthly, starting from the midpoints of the longest paths, it traverses midpoints of shorter and shorter paths and upon reaching midpoints of edges, it thaws synchronisation of the respective edges; sixthly, all edges finish synchronisation at time $r + d$ with the creation of fire signals that lie dense in the graph, where $r$ is the radius of the graph with respect to the general and $d$ is the diameter of the graph.

That the first task is performed is evident from the definitions of $\delta_e$, $\delta_v$, $\delta_{v\text{irtualTree}}$, and $\delta_{v\text{irtualTree}}$. The only subtlety here is that besides leaf signals there cannot be stationary signals in virtual leaves or, more precisely, at points that are virtually cut by leaf signals, because stationary signals carry the semi-direction vry, which is insufficient to
associate them with one or the other leaf signal as is done for non-
stationary signals. This is no problem because the other tasks do not
place stationary signals in (virtual) leaves. Therefore, we assume from
now on, without loss of generality, that the multigraph $\mathcal{G}$ is a tree.

That the second task is performed can be seen from a careful examina-
tion of the definitions of $\delta_{e,1}^{\text{tree}}, \delta_{v,1}^{\text{tree}}, \delta_{e,2}^{\text{tree}}, \delta_{v,2}^{\text{tree}}$, where from
the third task it is used that midpoints of edges are found in time to
start the freezing process.

That the third, fourth, and fifth and sixth parts are performed is
proven in section 6.1, section 6.2, and section 6.3.

6.1 Midpoints are Determined

The midpoint of a path in a multigraph is the midpoint of its embedding
in the continuum representation of the multigraph as introduced in

**Definition 6.1.** Let $p$ be a path in $\mathcal{G}$. The point $m_p = p(\omega(p)/2)$ is
called midpoint of $p$.

**Remark 6.2.** The midpoint of the empty path in $v$ is the vertex $v$
itself.

**Remark 6.3.** Let $p, q$, and $q'$ be three paths such that $\tau(p) = \sigma(q) =
\sigma(q')$, $m_{p \bullet q} \in p$, and $\omega(q) \geq \omega(q')$. Then, $m_{p \bullet q'} \in p$ and

$$d(m_{p \bullet q}, m_{p \bullet q'}) = \omega(p \bullet q)/2 - \omega(p \bullet q')/2 = \omega(q)/2 - \omega(q')/2.$$

Analogously, let $q$, $q'$, and $p$ be three paths such that $\tau(q) = \tau(q') =
\sigma(p)$, $m_{q \bullet p} \in p$, and $\omega(q) \geq \omega(q')$. Then, $m_{q' \bullet p} \in p$ and

$$d(m_{q \bullet p}, m_{q' \bullet p}) = \omega(q \bullet p)/2 - \omega(q' \bullet p)/2 = \omega(q)/2 - \omega(q')/2.$$  

When the midpoints of non-empty direction-preserving paths are
found is stated in

**Lemma 6.4.** Let $p$ be a non-empty direction-preserving path in $\mathcal{G}$. The
midpoint signal that designates the midpoint of $p$ is created at $m_p$ at
time $t_p = \max\{d(g, \sigma(p)), d(g, \tau(p))\} + \omega(p)/2$.  

**Proof Sketch.** First, let the general $g$ be the source or target of $p$.
Then, a find-midpoint signal with origin $g$ of speed 1 and a slowed-down
find-midpoint signal with origin $g$ of speed $1/3$ travel from $g$ towards
the other end of $p$. The find-midpoint signal is reflected at the other end
at time $(2/2) \cdot \omega(p)$ and this reflection collides with the slowed-down
find-midpoint signal at the midpoint of $p$ at time $(3/2) \cdot \omega(p)$ creating
a midpoint signal for $p$ (because both signals have the same origin).
Note that the time of collision is equal to $t_p$ (see figure 5.1a).
Secondly, let the general $g$ lie on $p$ without being its source or target. Then, two find-midpoint signals with origin $g$ travel from $g$ to the ends of $p$; the source of $p$ is reached at time $d(g, \sigma(p))$ and the target at time $d(g, \tau(p))$. When such a signal reaches its end, it is reflected and travels back. If $d(g, \sigma(p)) = d(g, \tau(p))$, then this distance is equal to $\omega(p)/2$, the reflected signals collide at time $\omega(p)$ at the midpoint of $p$ creating a midpoint signal for $p$; note that the time of collision is equal to $t_p$. Otherwise, the reflected signal that is nearer to $g$ reaches this vertex first, where the signal is slowed down and travels towards the other reflected signal with which it collides at the midpoint of $p$ at time $t_p$ (see figure 5.1b).

Lastly, let the general $g$ not lie on $p$. Then, an initiate signal travels from $g$ to the nearest vertex $v$ on $p$, where it creates two find-midpoint signals with origin $v$ that travel to the ends of $p$, are reflected at these ends and travel back, one is slowed-down upon reaching $v$, and the slowed-down signal collides with the reflected signal in the midpoint of $p$ at time $d(g, v) + \max\{d(v, \sigma(p)), d(v, \tau(p))\} + \omega(p)/2$ creating a midpoint signal for $p$. Note that the time of collision is equal to $t_p$. $\blacksquare$

**Remark 6.5.** When reflected/slowed-down find-midpoint signals with different origins collide, nothing happens, the signals just move on. Hence, no points are falsely found to be midpoints. □

That the midpoints of maximum-weight direction-preserving paths are identical and found at time $r + d$ is shown in

**Lemma 6.6.** All maximum-weight direction-preserving paths in $G$ have the same midpoint $\hat{m}$ and the midpoint signals that designate the midpoints of such paths are created at $\hat{m}$ at time $r + d/2$, where $r = \max_{v \in V} d(g, v)$ is the radius of $G$ with respect to $g$ and $d = \max_{v, v' \in V} d(v, v')$ is the diameter of $G$. Note that $r$ is equal to the radius $\sup_{m \in M} d(g, m)$ of $M$ with respect to $g$ and $d$ is equal to the diameter $\sup_{m, m' \in M} d(m, m')$ of $M$. $\blacksquare$

**Proof.** We prove both statements by contradiction.

First, suppose that there are two maximum-weight direction-preserving paths $\hat{p}$ and $\hat{p}'$ in $G$ that do not have the same midpoint. Then, there is a non-empty direction-preserving path $p_{m}$ in $M$ from the midpoint of $\hat{p}$ to the one of $\hat{p}'$. And, there is a direction-preserving subpath $p$ of $\hat{p}$ in $M$ from one end of $\hat{p}$ to its midpoint whose target-direction is not the reverse of the source-direction of $p_{m}$. And, there is a direction-preserving subpath $p'$ of $\hat{p}'$ in $M$ from the midpoint of $\hat{p}'$ to one of its ends whose source-direction is not the reverse of the target-direction of $p_{m}$. The concatenation of $p$, $p_{m}$, and $p'$ is a direction-preserving path from vertex to vertex in $M$ whose length is equal to $d/2 + \omega(p_{m}) + d/2 > d$. It corresponds to a direction-preserving path $p$ in $G$ whose

63
weight is greater than \( d \), which contradicts that \( d \) is the diameter of \( \mathcal{G} \). Therefore, all maximum-weight direction-preserving paths in \( \mathcal{G} \) have the same midpoint, which we denote by \( \hat{m} \).

Secondly, suppose that there is a maximum-weight direction-preserving path \( \hat{p} \) in \( \mathcal{G} \) such that \( \max\{d(g, \sigma(\hat{p})), d(g, \tau(\hat{p}))\} < r \). Let \( v \) be a vertex of \( \mathcal{G} \) such that \( d(g, v) = r \), let \( p_v \), \( p_\sigma \), and \( p_r \) be the direction-preserving paths in \( \mathcal{G} \) from \( v \), \( \sigma(\hat{p}) \), and \( \tau(\hat{p}) \) to \( g \), let \( v' \) be the vertex on \( p_v \) and \( p_\sigma \) or on \( p_v \) and \( p_r \) that is the furthest from \( g \), and let \( v'' \) be the vertex on \( \hat{p} \) that is the nearest to \( g \). Then, the weight of \( p_v \) is \( r \), the one of \( p_\sigma \) is \( d(g, \sigma(\hat{p})) < r \), and the one of \( p_r \) is \( d(g, \tau(\hat{p})) < r \). If \( v' \) lies on the subpath of \( p_\sigma \) from \( \sigma(\hat{p}) \) to \( v'' \) (which is equal to the subpath of \( \hat{p} \) with the same ends), then let \( p \) be the direction-preserving path from \( v \) over \( v' \) over \( v'' \) to \( \tau(\hat{p}) \); if \( v' \) lies on the subpath of \( p_r \) from \( \tau(\hat{p}) \) to \( v'' \) (which is equal to the subpath of the inverse of \( \hat{p} \) with the same ends), then let \( p \) be the direction-preserving path from \( v \) over \( v' \) over \( v'' \) to \( \sigma(\hat{p}) \); and otherwise, let \( p \) be the direction-preserving path from \( v \) over \( v' \) over \( v'' \) to \( \tau(\hat{p}) \) (we could have chosen \( \sigma(\hat{p}) \) as well). See figure 6.1 for a schematic representation of the three cases.

In the first case, because the subpaths of \( p_v \) and \( p_\sigma \) from \( v'' \) to \( g \) coincide and the weight of \( p_v \) is greater than the weight of \( p_\sigma \), the weight of the subpath of \( p_v \) from \( v \) over \( v' \) to \( v'' \) (which is equal to the subpath of \( p \) from \( v \) over \( v' \) to \( v'' \)) is greater than the weight of the subpath of \( p_\sigma \) from \( \sigma(\hat{p}) \) over \( v' \) to \( v'' \) (which is equal to the subpath of \( \hat{p} \) from \( \sigma(\hat{p}) \) over \( v' \) to \( v'' \)) and hence, because the subpaths of \( p \) and \( \hat{p} \) from \( v'' \) to \( \tau(\hat{p}) \) coincide, the weight of \( p \) is greater than the weight of \( \hat{p} \). In the second case, it follows analogously that the weight of \( p \) is greater than the weight of \( \hat{p} \). And in the third case, because the subpaths of \( p_v \) and \( p_\sigma \) from \( v' \) to \( g \) coincide and the weight of \( p_v \) is greater than the weight of \( p_\sigma \), the weight of the subpath of \( p_v \) from \( v \) to \( v' \) is greater than the weight of the subpath of \( p_\sigma \) from \( \sigma(\hat{p}) \) over \( v'' \) to \( v' \), hence the weight of the subpath of \( p \) from \( v \) over \( v' \) to \( v'' \) is greater than the
weight of the subpath of \( \hat{p} \) from \( \sigma(\hat{p}) \) to \( v'' \), and therefore, the weight of \( p \) is greater than the weight of \( \hat{p} \).

In either case, the inequality \( \omega(p) > \omega(\hat{p}) \) contradicts that \( \hat{p} \) is a maximum-weight path. Therefore, for each maximum-weight direction-preserving path in \( \mathcal{G} \), we have \( \max\{d(\mathbf{g}, \sigma(\hat{p})), d(\mathbf{g}, \tau(\hat{p}))\} = r \). It follows from lemma 6.4 that the midpoint signals that designate the midpoints of maximum-weight direction-preserving paths in \( \mathcal{G} \) are created at \( \mathbf{m} \) at time \( r + d/2 \). ■

It follows that the midpoints of non-empty non-maximum-weight direction-preserving paths are found before the ones of maximum-weight direction-preserving paths as shown in

**Corollary 6.7.** Let \( p \) be a non-empty direction-preserving path in \( \mathcal{G} \). The midpoint signal that designates the midpoint of \( p \) is created at \( \mathbf{m}_p \) before time \( r + d/2 \).

**Proof.** This is a direct consequence of lemmata 6.4 and 6.6, because \( d(\mathbf{g}, \sigma(p)) \leq r \), \( d(\mathbf{g}, \tau(p)) \leq r \), and \( \omega(p) < d \). ■

### 6.2 Midpoints of Maximum-Weight Paths are Recognised as Such

That the midpoints of maximum-weight direction-preserving paths are recognised as such is sketched in

**Remark 6.8.** The first two reflected find-midpoint signals, or the first two reflected and slowed-down find-midpoint signals to collide that originated at the same vertex and were reflected at the ends of a maximum-weight direction-preserving path are marked at the time of collision and hence recognise that the midpoint of the path they collide at is the one of a maximum-weight direction-preserving path. ■

**Proof Sketch.** Let \( \hat{p} \) be a maximum-weight direction-preserving path in \( \mathcal{G} \) (see figure 6.2). Then, the ends \( \hat{v}_1 \) and \( \hat{v}_2 \) of \( \hat{p} \) are leaves. And, among the first find-midpoint signals to reach the ends of \( \hat{p} \) are the two that originated at the vertex \( \hat{v} \) on \( \hat{p} \) that is nearest to \( \mathbf{g} \), and, because they travel alongside initiate signals, their reflections \( s \) and \( s' \) at the ends of \( \hat{p} \) are marked. When one of them reaches a vertex on its way back it stays marked, because from all directions excluding from the direction it is headed but including the direction it is coming from have the marked reflected find-midpoint signals that originated at the vertex just or already returned, which is memorised by a count signal that is located at the vertex; the reason that such marked signals have already returned is that otherwise there would be a direction-preserving path with more weight than \( \hat{p} \) that would be the concatenation the maximal subpath of \( \hat{p} \) that lies in the direction the signal is headed and a path.
that begins with one of the edges from which no marked signal has returned yet. Note that the signals \( s \) and \( s' \) travel back alongside the marked reflected find-midpoint signals that originated at the vertices the signals \( s \) and \( s' \) passed by before they were reflected and that therefore, when they reach vertices on their way back, the count signals are just updated and hence up-to-date.

When \( s \) and \( s' \) reach the vertex they originated at at the same time, then this vertex is found to be the midpoint of the path \( \hat{p} \) and, because \( s \) and \( s' \) are marked, it is recognised as the midpoint of a maximum-weight path. When one of the signals, say \( s \), reaches the vertex it originated at, namely \( \hat{v} \), first, then it is slowed down and spreads throughout the graph away from the edge it comes from, in particular, one slowed-down find-midpoint signal, let us denote it by \( s'' \), travels towards \( s' \). When \( s'' \) reaches a vertex on its way towards \( s' \) it stays marked, because from all directions excluding from the direction it is headed have the slowest reflected find-midpoint signals that originated at the vertex just or already returned, more precisely, from all directions excluding from the direction it is coming from and the one it is headed have the marked reflected find-midpoint signals that originated at the vertex just or already returned and from the direction it is coming from has the slowest but unmarked reflected find-midpoint signal that originated at the vertex and the first marked slowed-down find-midpoint signal just returned. When \( s'' \) and \( s' \) collide, the midpoint of \( \hat{p} \) is found
and, because $s''$ and $s'$ are marked, it is recognised as the midpoint of a maximum-weight path.

A detailed proof is given in the remainder of the present subsection. ■

Symbolic notations for the predicates is a vertex of and is an edge of are introduced in

**Definition 6.9.** Let $T = (W, F)$ be a subtree of $G$, let $v$ be a vertex of $G$, and let $e$ be an edge of $G$. We write $v \in T$ instead of $v \in W$ and $e \in T$ instead of $e \in F$.

The set of greatest subtrees of a vertex $v$ that correspond to its incident edges is named in

**Definition 6.10.** Let $v$ be a vertex of $G$.

a. Let $E_v$ be the set of edges that are incident to $v$, and, for each edge $e \in E_v$, let $T_{v,e}$ be the greatest subtree of $G$ that is rooted at $v$, contains the edge $e$, and does not contain any other edge of $E_v$ (note that by greatest we mean greatest with respect to the number of vertices). The set $\{T_{v,e} \mid e \in E_v\}$ is denoted by $sbtrs(v)$.

b. The set

$$\max sbtrs(v) = \arg \max_{T \in sbtrs(v)} \|T\|_v$$

is the set of maximum-radius trees of $sbtrs(v)$. In the case that it is a singleton set, we denote its one and only element by $\hat{T}_v$.

c. The set

$$\overline{sbtrs}(v) = \arg \max_{T \in sbtrs(v) \setminus \max sbtrs(v)} \|T\|_v$$

is the set of second-maximum-radius trees of $sbtrs(v)$. □

**Remark 6.11.** For each leaf $v$ of $G$, the set $\max sbtrs(v)$ is a singleton and it is equal to $sbtrs(v)$, and the set $\overline{sbtrs}(v)$ is empty. □

Things associated with a greatest subtree of a vertex are introduced in

**Definition 6.12.** Let $v$ be a vertex of $G$ and let $T$ be a tree of $sbtrs(v)$.

a. Let $e$ be the edge of $T$ that is incident to $v$. The direction that leads from $v$ onto $e$ is uniquely determined by $T$ and is denoted by $\text{dir}_v(T)$. □
The non-negative integer $\|T\|_v = \max_{v' \in T} d(v, v')$ is called \emph{radius of $T$ with respect to $v$}.

Let $P_T$ be the set of direction-preserving paths in $T$. The set

$$\text{max pths}_v(T) = \{ p \in P_T \mid \sigma(p) = v \text{ and } \omega(p) = \|T\|_v \}$$

is the set of maximum-weight direction-preserving paths from $v$ in $T$.

D. The set

$$\text{max vrtcs}_v(T) = \{ \tau(p) \mid p \in \text{max pths}_v(T) \}$$

is the set of vertices in $T$ that are furthest away from $v$. □

\textbf{Remark 6.13.} We have $|\text{sbtrs}(v)| = \deg(v)$ and $\text{dir}_v(\text{sbtrs}(v)) = \text{dir}(v)$. □

\textbf{Remark 6.14.} Each vertex of $\text{max vrtcs}_v(T)$ is a leaf. □

The unique direction-preserving path from one vertex to another is named in

\textbf{Definition 6.15.} Let $v$ and $v'$ be two vertices of $G$. The direction-preserving path in $G$ from $v$ to $v'$ is denoted by $p_{v,v'}$ and the vertices on this path are denoted by $V_{v,v'}$. □

When and why count signals, initiate signals, and (maybe-marked slowed-down/reflected) find-midpoint signals are created and how they spread throughout the tree is said in

\textbf{Remark 6.16.} Let the signal machine $\mathcal{S}$ be in the configuration $c_1$ at time 0.

A. At time 0, a count signal with empty memory is created at $g$, and initiate signals, find-midpoint signals with origin $g$, and maybe-marked slowed-down find-midpoint signals with origin $g$ and reflection vertex $g$ are sent from $g$ in all directions, where the slowed-down signal is marked if and only if $g$ is a leaf. For each vertex $v \in V \setminus \{g\}$, at time $d(g, v)$, an initiate signal reaches $v$ from the direction towards $g$, whereupon

\begin{itemize}
  \item[a)] a count signal with empty memory is created at $v$,
  \item[b)] initiate signals are sent from $v$ in all directions away from $g$, and
  \item[c)] find-midpoint signals with origin $v$, and maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v$ are sent from $v$ in all directions,
\end{itemize}
where the slowed-down signal is marked if and only if $v$ is a leaf.

In short, initiate signals spread from $g$ to all leaves, where they vanish, and they initiate the search for midpoints at all vertices. Note that to simplify the exposition of the forthcoming proofs, we also create count signals at leaves.

**B.** Let $v$ be a vertex of $G$. As said above, at time $d(g, v)$, find-midpoint signals with origin $v$, and maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v$ are sent from $v$ in all directions, where the slowed-down signal is marked if and only if $v$ is a leaf. For each vertex $v' \in V \setminus \{v\}$,

a) at time $d(g, v) + d(v, v')$, a find-midpoint signal with origin $v$ reaches $v'$ from the direction towards $v$, whereupon a maybe-marked reflected find-midpoint signal with origin $v$ and reflection vertex $v'$ is sent from $v'$ towards $v$ and find-midpoint signals with origin $v$ are sent from $v'$ in all directions away from $v$, where the reflected signal is marked if and only if $v$ is a leaf and an initiate signal just reached $v'$ (the latter is the case if and only if $d(g, v) + d(v, v') = d(g, v')$), and,

b) at time $d(g, v) + 3 \cdot d(v, v')$, a maybe-marked slowed-down find-midpoint signal with origin $v$ and reflection vertex $v$ reaches $v'$ from the direction towards $v$, whereupon maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v$ are sent from $v'$ in all directions away from $v$. Note that even if the slowed-down signal that is sent from $v$ at time $d(g, v)$ is marked, the slowed-down signal that reaches $v'$ may be unmarked, and even if the latter signal is marked, the slowed-down signals that are sent from $v'$ may be unmarked.

In short, find-midpoint signals with origin $v$ and maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v$ spread from $v$ to all leaves, and whenever one of the former signals reaches a vertex, it is also reflected. Note that to simplify the exposition of the forthcoming proofs, we talk as if maybe-marked slowed-down find-midpoint signals only vanish at leaves, although when a marked slowed-down find-midpoint signal collides with a marked reflected find-midpoint signal with the same origin, both signals vanish.

**C.** Let $v$ and $v'$ be two vertices of $G$ such that $v \neq v'$. As said above, at time $d(g, v) + d(v, v')$ a maybe-marked reflected find-midpoint signal with origin $v$ and reflection vertex $v'$ is sent from $v'$ towards $v$. For each vertex $w$ on the direction-preserving path from $v'$ to $v$ except for $v'$, at time $d(g, v) + d(v, v') + d(v', w)$, the signal reaches $w$ from the direction towards $v'$, whereupon,
if $w \neq v$, the signal is sent from $w$ towards $v$, and otherwise, maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v'$ are sent from $v$ in all directions away from $v'$.

And, for each vertex $v''$ of $G$ that from the viewpoint of $v$ lies in a direction away from $v'$, which means that $v'' \neq v$ and there is a tree $T \in \text{sbtrs}(v)$ such that $v' \notin T$ and $v'' \in T$, at time $d(g, v) + d(v, v') + d(v', v) + 3 \cdot d(v, v'')$, a maybe-marked slowed-down find-midpoint signal with origin $v$ and reflection vertex $v'$ reaches $v''$, whereupon maybe-marked slowed-down find-midpoint signals with origin $v$ and reflection vertex $v'$ are sent from $v''$ in all directions away from $v$ (or, equivalently, away from $v'$).

In short, each maybe-marked reflected find-midpoint signal travels back to its origin and when it reaches its origin, it is slowed-down and spreads to all leaves away from its reflection vertex. Note that unmarked signals never become marked, but marked signals may become unmarked; precisely when the latter does or does not happen is answered in the present subsection.

What is said above is evident from the definition of the signal machine $S$ (if it is carefully studied). □

When and why leaves sent marked reflected/slowed-down find-midpoint signals is said in

**Lemma 6.17.** Let $v$ be a leaf of $G$. At time $d(g, v)$, an initiate signal reaches $v$, for each vertex $w \in V_{g,v} \setminus \{v\}$, a find-midpoint signal with origin $w$ reaches $v$, and no other find-midpoint signal reaches $v$, whereupon

A. for each vertex $w \in V_{g,v} \setminus \{v\}$, a marked reflected find-midpoint signal with origin $w$ and reflection vertex $v$ is sent from $v$ towards $w$, which is the only possible direction,

B. a marked slowed-down find-midpoint signal with origin $v$ and reflection vertex $v$ is sent from $v$ towards $w$, and

C. no other marked signal is sent from $v$.

And before time $d(g, v)$ no signals reach and are sent from $v$, and after time $d(g, v)$ no marked signals are sent from $v$ (because after this time no initiate signal reaches $v$). □

**Proof.** This is a direct consequence of remark 6.16 and the definition of $S$. ■

When does a signal that is sent from a vertex towards the general at a special time reach the next vertex is answered in
Lemma 6.18. Let \( v \) be a vertex of \( G \), let \( T \) be a tree of \( \text{sbtrs}(v) \setminus \text{max sbtrs}(v) \) such that either \( v = g \) or \( g \notin T \), and let \( v' \) be the one and only neighbour of \( v \) in \( T \). Then,

A. \( \text{max sbtrs}(v') \) is a singleton set and its only element \( \hat{T}_{v'} \) contains \( v \),

B. \( \text{max vrtcs}_v(T) = \begin{cases} \{v'\}, & \text{if } v' \text{ is a leaf,} \\ \bigcup_{T' \in \text{sbtrs}(v')} \text{max vrtcs}_{v'}(T'), & \text{otherwise,} \end{cases} \)

C. \( \|T\|_v = d(v, v') + \max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'} \), and

D. when a signal of speed 1 is sent from \( v' \) towards \( v \) at time \( d(g, v') \) + \( 2 \cdot \max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'} \), it reaches \( v \) at time \( d(g, v) + 2 \cdot \|T\|_v \), where in the case that \( \text{sbtrs}(v') \) is empty, we define \( \max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'} \) as 0.

Proof. The first item is evident, the second and third follow from it, and the fourth follows from the third with \( d(g, v') = d(g, v) + d(v, v') \) and the fact that the signal needs the time span \( d(v, v') \) to traverse the edge from \( v' \) to \( v \).

The set of all non-leaf vertices whose unique maximum-radius tree contains the general is named in

Definition 6.19. The set of all non-leaf vertices \( v \) of \( G \) such that \( \text{max sbtrs}(v) \) is a singleton set and its only element \( \hat{T}_v \) contains the vertex \( g \), is denoted by \( V_g \).

Remark 6.20. For each vertex \( v \in V_g \), each tree \( T \in \text{sbtrs}(v) \setminus \{\hat{T}_v\} \), and each vertex \( v' \) of \( T \), we have \( v' \in V_g \). And, for each vertex \( v \in V_g \), the set \( \text{sbtrs}(v) \) is non-empty and, for each tree \( T \in \text{sbtrs}(v) \), we have \( \|T\|_v = \max_{T' \in \text{sbtrs}(v)} \|T'\|_v \).

When and why non-leaf vertices whose unique maximum-radius tree contains the general sent marked reflected/slowed-down find-midpoint signals is shown in

Lemma 6.21. For each vertex \( v \in V_g \), at time \( d(g, v) + 2 \cdot \max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'} \),

A. the count signal at \( v \), before it is updated, has the memory \( \text{dir}_v(\text{sbtrs}(v) \cup \{\hat{T}_v\}) \),

B. for each tree \( T \in \text{sbtrs}(v) \), each leaf \( \bar{v} \in \text{max vrtcs}_v(T) \), and each vertex \( w \in V_g \), a marked reflected find-midpoint signal with origin \( w \) and reflection vertex \( \bar{v} \) reaches \( v \) from direction \( \text{dir}_v(T) \), and
C. no other marked reflected find-midpoint signal reaches v,

whereupon

A. the count signal at v, after it is updated, has the memory \( \text{dir}(v) \setminus \{ \text{dir}_v(\hat{T}_v) \} \),

B. for each tree \( T \in \overline{sbrts}(v) \), each leaf \( \bar{v} \in \max \text{vrtcs}_v(T) \), and each vertex \( w \in V_{\bar{v}, v} \setminus \{ v \} \), a marked reflected find-midpoint signal with origin \( w \) and reflection vertex \( \bar{v} \) is sent from \( v \) in the direction \( \text{dir}_v(\hat{T}_v) \) towards \( w \), and no other marked reflected find-midpoint signal with origin \( w \) and reflection vertex \( \bar{v} \) is sent from \( v \), and,

C. for each tree \( T \in \overline{sbrts}(v) \), each leaf \( \bar{v} \in \max \text{vrtcs}_v(T) \), and each direction \( d \in \text{dir}(v) \setminus \{ \text{dir}_v(T) \} \), a marked slowed-down find-midpoint signal with origin \( v \) and reflection vertex \( \bar{v} \) is sent from \( v \) in direction \( d \) (note that \( \text{dir}_v(\hat{T}_v) \in \text{dir}(v) \setminus \{ \text{dir}_v(T) \} \)), and, no other marked slowed-down find-midpoint signal with origin \( v \) and reflection vertex \( \bar{v} \) is sent from \( v \).

And before time \( d(\bar{g}, v) + 2 \cdot \max_{T \in \overline{sbrts}(v)} \| T \|_v \), marked signals may reach \( v \) but no marked signal is sent from \( v \), and after that time, no marked signals as above are sent from \( v \).

**Proof.** We prove this by induction on \( n_v = \max \{ W,F \in sbrts(v) \setminus \{ \hat{T}_v \} \} |F| \), for \( v \in V_{\bar{g}} \). For brevity though, we only treat the existence of signals and not their absence.

**Base Case (see figure 6.3).** Let \( v \in V_\bar{g} \) such that \( n_v = 1 \). And, let \( T \in \overline{sbrts}(v) \setminus \{ \hat{T}_v \} \). Then, \( T \) consists of one edge \( e \) whose one end is \( v \), whose other end is a leaf \( v' \), and whose weight is \( \| T \|_v \). According to lemma 6.17, at time \( d(\bar{g}, v') = d(\bar{g}, v) + \| T \|_v \), for each vertex \( w \in V_{\bar{g}, v} \setminus \{ v' \} = V_{\bar{g}, v} \), a marked reflected find-midpoint signal with origin \( w \) and reflection vertex \( v' \) is sent from \( v' \) towards \( v \), in particular, one with origin \( v \). These signals reach \( v \) at time \( d(\bar{g}, v) + 2 \cdot \| T \|_v \), whereupon

A. the count signal at \( v \) memorises the direction \( \text{dir}_v(T) \) (because a marked reflected find-midpoint signal with origin \( v \) reached \( v \)),

B. for each vertex \( w \in V_{\bar{g}, v} \setminus \{ v \} \), a maybe-marked reflected find-midpoint signal with origin \( w \) and reflection vertex \( v' \) is sent from \( v \) in the direction \( \text{dir}_v(\hat{T}_v) \) towards \( w \), and

C. for each direction \( d \in \text{dir}(v) \setminus \{ \text{dir}_v(T) \} \), a maybe-marked slowed-down find-midpoint signal with origin \( v \) and reflection vertex \( v' \) is sent from \( v \) in direction \( d \).

On the timeline, for the trees of \( \overline{sbrts}(v) \setminus \{ \hat{T}_v \} \) in non-decreasing order with respect to the radius and at the same time for trees with the same radius, the signals reach \( v \) and are sent from \( v \). For those trees
whose radius is less than the second greatest radius among the trees of $\text{sbtrs}(v)$, which is $\max_{T \in \text{sbtrs}(v)} \|T\|_v$, the aforementioned maybe-marked signals that are sent from $v$ are unmarked (because the memory of the count signal, after it is updated, does neither contain the directions of $\text{dir}_v(\text{sbtrs}(v))$ nor the direction $\text{dir}_v(\hat{T}_v)$). And, for the trees of $\text{sbtrs}(v)$, the aforementioned maybe-marked signals that are sent from $v$ are marked (because the count signal, after it is updated, has the memory $\text{dir}_v(\text{max vrtcs}(T)) \setminus \{\text{dir}_v(\hat{T}_v)\}$). In conclusion, at time $d(g,v) + 2 \cdot \max_{T \in \text{sbtrs}(v)} \|T\|_v$, what is to be proven holds.

**Inductive Step (see figure 6.4).** Let $v \in V_g$ such that $n_v \geq 2$ and such that what is to be proven holds for each vertex $v' \in V_g$ with $n_{v'} < n_v$, which is the **inductive hypothesis**. And, let $T \in \text{sbtrs}(v) \setminus \{\hat{T}_v\}$, let $e$ be the edge of $T$ whose one end is $v$, and let $v'$ be the other end of $e$. The vertex $v'$ is either a leaf or an element of $V_g$ with $n_{v'} < n_v$.

In the first case, according to lemma 6.17, at time $d(g,v') = d(g,v) + d(v,v')$, for each vertex $w \in V_{g,v'} \setminus \{v'\} = V_{g,v}$, a marked reflected find-midpoint signal with origin $w$ and reflection vertex $v'$ is sent from $v'$ towards $v$; note that $v'$ is the one and only element of $\text{max vrtcs}_v(T)$, the set $\text{sbtrs}(v')$ is empty, and we define $\max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'}$ as 0 (see lemma 6.18). In the second case, according to the inductive hypothesis, at time $d(g,v') + 2 \cdot \max_{T' \in \text{sbtrs}(v')} \|T'\|_{v'}$, for each tree $T' \in \text{sbtrs}(v')$, each leaf $\hat{v}' \in \text{max vrtcs}_{v'}(T')$, and each vertex $w \in V_{g,v} \setminus \{v'\} = V_{g,v}$, a marked reflected find-midpoint signal with origin $w$ and reflection
vertex $v'$ is sent from $v'$ towards $v$; note that the set of all vertices $v'$ is equal to $\text{max vrtcs}_v(T)$ (see lemma 6.18).

In both cases, the marked signals reach $v$ at time $d(g, v) + d(v, v') + 2 \cdot \max_{T' \in \text{max sbtrs}(v')}\|T'\|_v + d(v', v)$, which is equal to $d(g, v) + 2 \cdot \|T\|_v$.

whereupon

A. the count signal at $v$ memorises the direction $\text{dir}_v(T)$ (because at least one marked reflected find-midpoint signal with origin $v$ reached $v$),

B. for each leaf $\tilde{v} \in \text{max vrtcs}_v(T)$ and each vertex $w \in V_{g,v} \setminus \{v\}$, a maybe-marked reflected find-midpoint signal with origin $w$ and reflection vertex $\tilde{v}$ is sent from $v$ in the direction $\text{dir}_v(\tilde{T}_v)$ towards $w$, and,

C. for each leaf $\tilde{v} \in \text{max vrtcs}_v(T)$ and each direction $d \in \text{dir}(v) \setminus \{\text{dir}_v(T)\}$, a maybe-marked slowed-down find-midpoint signal with origin $v$ and reflection vertex $\tilde{v}$ is sent from $v$ in direction $d$.

It follows verbatim as in the base case, that what is to be proven holds. 

The set of all vertices whose unique maximum-radius tree does not contain the general is named in

\textsc{Definition 6.22.} The set of all vertices $v$ of $G$ such that $\text{max sbtrs}(v)$ is a singleton set and its only element $\tilde{T}_v$ does not contain the vertex $g$, is denoted by $U_g$. \hfill \Box

\textsc{Remark 6.23.} We have $g \notin U_g$. And, for each vertex $v \in U_g$, we have $V_{g,v} \setminus \{g\} \in U_g$, the vertex $v$ is not a leaf and the set $\text{sbtrs}(v)$ is non-empty. \hfill \Box

The maximal subtree of a non-general vertex that contains the general is named in

\textsc{Definition 6.24.} Let $v$ be a vertex of $G$ such that $v \neq g$. The tree of $\text{sbtrs}(v)$ that does contain the vertex $g$ is denoted by $T^g_v$. And to avoid case differentiations, we define $T^g_g$ as the number 0. \hfill \Box

The vertices of the maximal path from a vertex to the general whose vertices excluding its source have the subtree that contains the general as second-maximum-radius subtree is named in

\textsc{Definition 6.25.} Let $v$ be a vertex of $U_g$, and let $p$ be the maximum-weight subpath of $p_{g,v}$ such that $\tau(p) = v$ and, for each vertex $w$ on $p$ with $w \neq \sigma(p)$, we have $T^g_w \in \text{sbtrs}(w)$. The set of the vertices on $p$ is denoted by $W_{g,v}$. See figure 6.5. \hfill \Box
Figure 6.5: Schematic representation of $G$, where $v \in U_g$, $W_{g,v} = \{v''', w', w, v\}$, $W_{g,w} = \{w'', w', w\}$, $W_{g,w'} = \{w'', w'\}$, $W_{g,w''} = \{w''\}$, and subtrees of vertices and their radii are depicted as dashed triangles of various sizes.

**Remark 6.26.** We have $v \in W_{g,v} \setminus \{g\} \subseteq U_g$. And, for each $w \in W_{g,w}$, we have $W_{g,w} \subseteq W_{g,v}$. And, if $\mathcal{T}_w^g \in \text{sbrts}(v)$, then $W_{g,w} = W_{g,w'} \cup \{v\}$, where $w'$ is the neighbour of $v$ on $p_{g,v}$. □

When and why vertices whose unique maximum-radius tree does not contain the general sent marked reflected/slowed-down find-midpoint signals is shown in

**Lemma 6.27.** For each vertex $v \in U_g$, at time $d(g,v) + 2 \cdot \max_{T \in \text{sbrts}(v)} \|T\|_v$,

A. the count signal at $v$, before it is updated, has the memory $\text{dir}(v) \setminus \text{dir}_v(\text{sbrts}(v) \cup \{\mathcal{T}_v\})$,  

B. for each tree $T \in \text{sbrts}(v)$ and each leaf $\bar{v} \in \max \text{vrtcs}_v(T)$, a maybe-marked reflected find-midpoint signal with origin $v$ and reflection vertex $\bar{v}$ reaches $v$ from direction $\text{dir}_v(T)$, where the reflected signal is marked if and only if $T \neq \mathcal{T}_v^g$, and,

C. for each vertex $w \in W_{g,w} \setminus \{v\}$, each tree $T \in \text{sbrts}(w) \setminus \{\mathcal{T}_w^g\}$, and each leaf $\bar{w} \in \max \text{vrtcs}_w(T)$, a marked slowed-down find-midpoint signal with origin $w$ and reflection vertex $\bar{w}$ reaches $v$ from the direction towards $w$, which is the direction $\text{dir}_v(\mathcal{T}_w^g)$,

whereupon

A. the count signal at $v$, after it is updated, has the memory $\text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T}_v)\}$,

B. for each tree $T \in \text{sbrts}(v) \setminus \{\mathcal{T}_v^g\}$, each leaf $\bar{v} \in \max \text{vrtcs}_v(T)$, and each direction $d \in \text{dir}(v) \setminus \{\text{dir}_v(T)\}$, a marked slowed-down find-midpoint signal with origin $v$ and reflection vertex $\bar{v}$ is sent from $v$ in direction $d$ (note that $\{\text{dir}_v(\mathcal{T}_v), \text{dir}_v(\mathcal{T}_w^g)\} \subseteq \text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T}_v)\}$), and,

C. for each vertex $w \in W_{g,w} \setminus \{v\}$, each tree $T \in \text{sbrts}(w) \setminus \{\mathcal{T}_w^g\}$, each leaf $\bar{w} \in \max \text{vrtcs}_w(T)$, and each direction $d \in \text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T}_w^g)\}$, a marked slowed-down find-midpoint signal with origin $w$ and reflection vertex $\bar{w}$ is sent from $v$ in direction $d$ (note
that \( \text{dir}_v(\mathcal{T}_v) \in \text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T}_v^g)\} \) and that, if \( w = g \), then \( \text{sbtrs}(w) \setminus \{\mathcal{T}_w^g\} = \text{sbtrs}(w) \).

\[\Box\]

**Proof.** We prove this by induction on \( n_v = |V_{g,v}| \), for \( v \in U_g \).

**Base Case (compare figure 6.6).** Let \( v \in U_g \) such that \( n_v = 1 \).

First, for each tree \( \mathcal{T} \in \text{sbtrs}(v) \setminus \{\mathcal{T}_v^g, \hat{T}_v\} \), according to lemma 6.17, if the neighbour of \( v \) in \( \mathcal{T} \) is a leaf, or lemma 6.21, otherwise, and lemma 6.18, at time \( d(g,v) + 2 \cdot ||\mathcal{T}\||_v \), for each leaf \( \tilde{v} \in \max \text{vrtcs}_v(\mathcal{T}) \), a marked reflected find-midpoint signal with origin \( v \) and reflection vertex \( \tilde{v} \) reaches \( v \) from direction \( \text{dir}_v(\mathcal{T}) \), whereupon

A. the count signal at \( v \) memorises the direction \( \text{dir}_v(\mathcal{T}) \),

B. for each leaf \( \tilde{v} \in \max \text{vrtcs}_v(\mathcal{T}) \) and each direction \( d \in \text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T})\} \), a maybe-marked slowed-down find-midpoint signal with origin \( v \) and reflection vertex \( \tilde{v} \) is sent from \( v \) in direction \( d \).

Secondly, according to remark 6.16, at time \( d(g,v) + 2 \cdot ||\mathcal{T}_v^g||_v \), for each leaf \( \tilde{v} \in \max \text{vrtcs}_v(\mathcal{T}_v^g) \), a maybe-marked reflected find-midpoint signal with origin \( v \) and reflection vertex \( \tilde{v} \) reaches \( v \) from direction \( \text{dir}_v(\mathcal{T}_v^g) \). This signal is actually unmarked, because it was reflected at \( \tilde{v} \) at time \( d(g,v) + ||\mathcal{T}_v^g||_v = d(g,v) + d(v,\tilde{v}) \), which, because \( v \neq g \) and \( \tilde{v} \in \mathcal{T}_v^g \), is greater than the only time, namely \( d(g,\tilde{v}) \), at which an initiate signal reaches \( \tilde{v} \).

Thirdly, because \( U_g \) is non-empty, we have \( g \in V_g \) and \( v \in \hat{T}_g \). According to lemma 6.21, at time \( 2 \cdot \max_{\mathcal{T} \in \text{sbtrs}(g)} ||\mathcal{T}\||_g \), for each tree \( \mathcal{T} \in \text{sbtrs}(g) \), and each leaf \( \tilde{g} \in \max \text{vrtcs}_g(\mathcal{T}) \), a marked slowed-down find-midpoint signal with origin \( g \) and reflection vertex \( \tilde{g} \) is sent from \( g \) towards \( v \); note that the set of all vertices \( \tilde{g} \) is equal to \( \max \text{vrtcs}_v(\mathcal{T}_v^g) \) (compare lemma 6.18). The marked signals reach \( v \) from the direction towards \( g \), which is the direction \( \text{dir}_v(\mathcal{T}_v^g) \), at time \( 2 \cdot \max_{\mathcal{T} \in \text{sbtrs}(g)} ||\mathcal{T}\||_g + 3 \cdot d(g,v) = d(g,v) + 2 \cdot ||\mathcal{T}_v^g||_v \), whereupon

A. the count signal at \( v \) memorises the direction \( \text{dir}_v(\mathcal{T}_v^g) \) and,

B. for each leaf \( \tilde{g} \in \max \text{vrtcs}_v(\mathcal{T}_v^g) \) and each direction \( d \in \text{dir}(v) \setminus \{\text{dir}_v(\mathcal{T}_v^g)\} \), a maybe-marked slowed-down find-midpoint signal with origin \( g \) and reflection vertex \( \tilde{g} \) is sent from \( v \) in direction \( d \).

Altogether, on the timeline, for the trees of \( \text{sbtrs}(v) \setminus \{\hat{T}_v\} \) in non-decreasing order with respect to the radius and at the same time for trees with the same radius, the signals reach \( v \) and are sent from \( v \). For those trees whose radius is less than the second greatest radius among the trees of \( \text{sbtrs}(v) \), which is \( \max_{\mathcal{T} \in \text{sbtrs}(v)} ||\mathcal{T}\||_v \), the aforementioned maybe-marked signals that are sent from \( v \) are unmarked (because the memory of the count signal, after it is updated, does neither contain the directions of \( \text{dir}_v(\text{sbtrs}(v)) \) nor the direction \( \text{dir}_v(\hat{T}_v) \)). And, for the trees of \( \text{sbtrs}(v) \), the aforementioned maybe-marked signals that are
sent from \( v \) are marked (because the count signal, after it is updated, has the memory \( \text{dir}(v) \setminus \{ \text{dir}_v(T_v') \} \)). Note that, if \( T'_v \in \text{subtrs}(v) \), then \( W_{g,v} \setminus \{ v \} = \{ g \} \), and otherwise, \( W_{g,v} \setminus \{ v \} = \emptyset \). In conclusion, at time \( d(g,v) + 2 \cdot \max_{T \in \text{subtrs}(v)} \| T \|_v \), what is to be proven holds.

**Inductive Step (see figure 6.6).** Let \( v \in U_g \) such that \( n_v \geq 2 \) and such that what is to be proven holds for each vertex \( v' \in U_g \) with \( n_{v'} < n_v \), which is the **inductive hypothesis**.

First, for each tree \( T \in \text{subtrs}(v) \setminus \{ T'_v, T_v' \} \), the same as in the base case happens.

Secondly, as in the base case, at time \( d(g,v) + 2 \cdot \| T'_v \|_v \), for each leaf \( v \) \( \vec{v} \in \max \text{vrtcs}_v(T'_v) \), an unmarked reflected find-midpoint signal with origin \( v \) and reflection vertex \( \vec{v} \) reaches \( v \) from direction \( \text{dir}_v(T'_v) \).

Thirdly, let \( v' \) be the neighbour of \( v \) in \( T'_v \). Then, \( v' \in V_{g,v} \setminus \{ g \} \subseteq U_g \) and \( n_{v'} < n_v \). According to the inductive hypothesis, at time \( d(g,v') + 2 \cdot \max_{T \in \text{subtrs}(v')} \| T' \|_{v'} \),

A. for each tree \( T' \in \text{subtrs}(v') \setminus \{ T'_v \} \) and each leaf \( \vec{v}' \in \max \text{vrtcs}_{v'}(T') \), a marked slowed-down find-midpoint signal with origin \( v' \) and reflection vertex \( \vec{v}' \) is sent from \( v' \) towards \( v \), and,

B. for each vertex \( w' \in W_{g,v} \setminus \{ v' \} \), each tree \( T' \in \text{subtrs}(w') \setminus \{ T'_v \} \), each leaf \( \vec{w}' \in \max \text{vrtcs}_{w'}(T') \), a marked slowed-down find-midpoint signal with origin \( w' \) and reflection vertex \( \vec{w}' \) is sent from \( v' \) towards \( v \).

The marked signals reach \( v \) from direction \( \text{dir}_v(T'_v) \) at time \( d(g,v') + 2 \cdot \max_{T \in \text{subtrs}(v')} \| T' \|_{v'} + 3 \cdot d(v',v) = d(g,v') + 2 \cdot \| T'_v \|_v \), whereupon

A. the count signal at \( v \) memorises the direction \( \text{dir}_v(T'_v) \) and,

B. for each vertex \( w \in W_{g,v} \), each tree \( T \in \text{subtrs}(w) \setminus \{ T'_v \} \), each leaf \( \vec{w} \in \max \text{vrtcs}_w(T) \), and each direction \( d \in \text{dir}(v) \setminus \{ \text{dir}_v(T'_v) \} \), a maybe-marked slowed-down find-midpoint signal with origin \( w \) and reflection vertex \( \vec{w} \) is sent from \( v \) in direction \( d \).

Note that, if \( T'_v \in \text{subtrs}(v) \), then \( W_{g,v} \setminus \{ v \} = W_{g,v'} \), and otherwise, \( W_{g,v} \setminus \{ v \} = \emptyset \). With that it follows verbatim as in the base case, that what is to be proven holds.
That midpoints of maximum-weight direction-preserving paths are recognised as such is shown in

**Theorem 6.28.** Let \( \hat{p} \) be a maximum-weight direction-preserving path in \( G \), let \( \hat{v} \) be the vertex on \( \hat{p} \) that is nearest to \( g \), let \( \hat{v}_1 \) and \( \hat{v}_2 \) be the two ends of \( \hat{p} \) such that \( d(\hat{v}, \hat{v}_1) \leq d(\hat{v}, \hat{v}_2) \), and let \( \hat{m} \) be the midpoint of \( \hat{p} \). At time \( r + d/2 \), at the midpoint \( \hat{m} \),

A. if \( d(\hat{v}, \hat{v}_1) = d(\hat{v}, \hat{v}_2) \), two marked reflected find-midpoint signals with origin \( \hat{v} \) and reflection vertices \( \hat{v}_1 \) and \( \hat{v}_2 \) collide, and

B. otherwise, a marked slowed-down find-midpoint signal with origin \( \hat{v} \) and reflection vertex \( \hat{v}_1 \) collides with a marked reflected find-midpoint signal with origin \( \hat{v} \) and reflection vertex \( \hat{v}_2 \).

**Proof Sketch.** From a broad perspective and ignoring boundary cases the following happens in the given order (see figure 6.7d). At time 0, an initiate signal is sent from \( g \) towards \( \hat{v} \). At time \( d(g, \hat{v}) \), this signal reaches \( \hat{v} \), whereupon find-midpoint signals with origin \( \hat{v} \) are sent from \( \hat{v} \) in all directions. At time \( d(g, \hat{v}) + \|T_\hat{v}\|_e \), the slowest but unmarked reflected find-midpoint signal with origin \( \hat{v} \) returns to \( \hat{v} \) from the direction towards \( g \). At time \( d(g, \hat{v}) + 2 \cdot d(\hat{v}, \hat{v}_1) \), the slowest and marked reflected find-midpoint signal with origin \( \hat{v} \) returns to \( \hat{v} \) from the direction towards \( \hat{v}_1 \), whereupon a marked slowed-down find-midpoint signal with origin \( \hat{v} \) is sent towards \( \hat{v}_2 \). At time \( d(g, \hat{v}) + 2 \cdot d(\hat{v}, \hat{v}_1) + 3 \cdot d(\hat{v}, \hat{m}) = d(g, \hat{m}) + 2 \cdot d(\hat{v}_1, \hat{m}) = r + d/2 \), this signal reaches \( \hat{m} \). And, at the same time, which is equal to \( d(g, \hat{v}) + d(\hat{v}, \hat{v}_2) + d(\hat{v}_2, \hat{m}) \), the slowest and marked reflected find-midpoint signal with origin \( \hat{v} \) that is on its way to return to \( \hat{v} \) from the direction towards \( \hat{v}_2 \) reaches \( \hat{m} \). The two marked signals collide at \( \hat{m} \) recognising it as the midpoint of a maximum-weight path.

The slowest reflected find-midpoint signal with origin \( \hat{v} \) that returns to \( \hat{v} \) from the direction towards \( g \) is unmarked, because it reaches the leaf it is reflected at later than the initiate signal and is never marked in the first place. The slowest reflected find-midpoint signal with origin \( \hat{v} \) that returns to \( \hat{v} \) from the direction towards \( \hat{v}_1 \) is marked, because it reaches the leaf \( \hat{v}_1 \) along with initiate signals and at each vertex on its way back, it is the penultimate marked signal to return (this is essentially lemma 6.21). For the same reason, the signal from the direction towards \( \hat{v}_2 \) is marked when it reaches \( \hat{m} \). And, the slowed-down find-midpoint signal with origin \( \hat{v} \) that reaches \( \hat{m} \) from the direction towards \( \hat{v} \) is marked, because at each vertex it reaches on its way, it is the penultimate marked signal to do so (this is essentially lemma 6.27). Note that, for each vertex \( u \) on the path from \( g \) to \( \hat{m} \) except for \( g \), the direction towards \( g \) is added to the memory of the count signal at \( u \), because a marked slowed-down find-midpoint signal whose origin is not \( u \) reaches \( u \) from that direction, and the other directions are added, because marked reflected find-midpoint signals with origin \( u \) reach it.
from those directions; for all other vertices, the latter is the reason the directions are added.

**Proof.** Let $v_1$ and $v_2$ be the neighbours of $\hat{m}$ on $\hat{p}$ such that $d(v_1, \hat{v}_1) < d(v_2, \hat{v}_2)$ (or, equivalently, $d(v_1, \hat{v}_2) > d(v_2, \hat{v}_2)$).

First, let $d(\hat{v}, \hat{v}_1) = d(\hat{v}, \hat{v}_2)$ (see figures 6.7a and 6.7c). Then, $\hat{v} = \hat{m}$, and, for each index $i \in \{1, 2\}$, we have $v_i \neq g$ (because, if $g$ lies on $\hat{p}$, then $g = \hat{v} = \hat{m} \neq v_i$, and otherwise, $g$ does not lie on $\hat{p}$ but $v_i$ does), and, $v_i$ is either a leaf or an element of $V_g$ (because $\text{max sbtrs}(v_i)$ consists of the tree of $\text{sbtrs}(v_i)$ that contains $\hat{v}_j$, where $j \in \{1, 2\} \setminus \{i\}$, and this tree, namely $\hat{T}_{v_i}$, contains $g$). Hence, according to lemma 6.17 or lemma 6.21, for each index $i \in \{1, 2\}$, at time $d(g, v_i) + 2 \cdot d(v_i, \hat{v}_i)$, a marked reflected find-midpoint signal with origin $\hat{m}$ and reflection vertex $\hat{v}_i$ is sent from $v_i$ towards $\hat{m}$, and at time $d(g, \hat{m}) + 2 \cdot d(\hat{m}, \hat{v}_i) = r + d/2$, it reaches $\hat{m}$, where it collides with the other signal.

Secondly, let $d(\hat{v}, \hat{v}_1) \neq d(\hat{v}, \hat{v}_2)$ (see figures 6.7b and 6.7d). Furthermore, let $X_g$ be the union of the set of leaves and the set $V_g$. Then, $\hat{v} \neq \hat{m}$, and either $v_1 = g = \hat{v} \in X_g$ or $v_1 \in U_g$, and $v_2 \in X_g$ (because otherwise, there would be a direction-preserving path with more weight than $\hat{p}$, which contradicts that $\hat{p}$ has maximum-weight).

According to lemma 6.17 or lemma 6.21, at time $d(g, v_1) + 2 \cdot d(v_1, \hat{v}_1)$, a marked reflected find-midpoint signal with origin $\hat{v}$ and reflection vertex $\hat{v}_1$ is sent from $v_1$ towards $\hat{m}$, and at time $d(g, v_2) + 2 \cdot d(v_2, \hat{v}_2) + d(v_2, \hat{m}) = r + d/2$, it reaches $\hat{m}$.

If $v_1 = g = \hat{v} \in X_g$, then, according to lemma 6.17 or lemma 6.21, at time $2 \cdot d(\hat{v}, \hat{v}_1)$ (which is 0 in the case that $v_1$ is a leaf, because in this case $v_1 = \hat{v} = \hat{v}_1$), a marked slowed-down find-midpoint signal with origin $\hat{v}$ and reflection vertex $\hat{v}_1$ is sent from $\hat{v}$ towards $\hat{m}$, and at time $2 \cdot d(\hat{v}, \hat{v}_1) + d(\hat{v}, \hat{m}) = r + d/2$, it reaches $\hat{m}$.

Otherwise, we have $\hat{v} \in W_{g,v_1} \setminus \{v_1\}$ (because, for each vertex $v \in V_{\hat{v},v_1} \setminus \{\hat{v}\}$, the set $\text{max sbtrs}(v)$ consists of the tree of $\text{sbtrs}(v)$ that contains $\hat{v}_1$, the tree $\hat{T}_v$ contains $\hat{v}_1$, and due to the maximality of $\hat{p}$, except for $\hat{T}_v$, there can be no tree of $\text{sbtrs}(v)$ with a greater radius than $\hat{T}_v$), and hence, according to lemma 6.27, at time $d(g, v_1) + 2 \cdot d(v_1, \hat{v}_1)$, a marked slowed-down find-midpoint signal with origin $\hat{v}$ and reflection vertex $\hat{v}_1$ is sent from $v_1$ towards $\hat{m}$, and at time $d(g, v_1) + 2 \cdot d(v_1, \hat{v}_1) + d(v_1, \hat{m}) = r + d/2$, it reaches $\hat{m}$.

In either case, at time $r + d/2$, at the midpoint $\hat{m}$, a marked slowed-down find-midpoint signal with origin $\hat{v}$ and reflection vertex $\hat{v}_1$ collides with a marked reflected find-midpoint signal with origin $\hat{v}$ and reflection vertex $\hat{v}_2$.

**Remark 6.29.** On the other hand, for each direction-preserving path that does not have maximum-weight, whenever its midpoint is found, one of the signals is unmarked and hence the midpoint is not falsely
Figure 6.7: Schematic representation of the set-up of the proof of theorem 6.28. Vertices are depicted as dots, points that may or may not be vertices are depicted as circles, paths that may consist of more than one edge are drawn dotted, and paths that consist of precisely one edge are drawn solid.
thought to be one of a maximum-weight path. The interested reader
may prove that for herself.

6.3 Thaw Signals Traverse Midpoints and Thaw Synchronisation of
Edges Just in Time

The inverse of a path is its traversal from target to source as given in

\textsc{Definition 6.30.} Let \( p = (v_0, e_1, v_1, \ldots, e_n, v_n) \) be a path in \( G \). The
path \( \text{inv}(p) = (v_n, e_n, \ldots, v_1, e_1, v_0) \) is called \textit{inverse} of \( p \).

\textsc{Remark 6.31.} The inverse of an empty path is the empty path itself.

\textsc{Remark 6.32.} The weight and the midpoint of the inverse of a path
is the same as the weight and the midpoint of the path itself.

An undirected path does not know which of its ends is its source and
which is its target and it can be represented as in

\textsc{Definition 6.33.} Let \( \leftrightarrow \) be the equivalence relation on \( P \) given by
\( p \leftrightarrow \text{inv}(p) \). Each equivalence class \( [p]_\leftrightarrow \in P / \leftrightarrow \) is called \textit{undirected path}
and the non-negative real number \( \omega([p]_\leftrightarrow) = \omega(p) \) is called \textit{weight}
of \( [p]_\leftrightarrow \).

\textsc{Remark 6.34.} The equivalence class of an empty path is the singleton
set that consists of the empty path.

For each path \( p \), the set(s) of paths with the same source (or target),
less weight, and midpoints on the continuum representation of \( p \) are
named in

\textsc{Definition 6.35.} For each direction-preserving path \( p \in P_\rightarrow \), let

\[ \Sigma_p = \{ p' \in P_\rightarrow \mid \sigma(p') = \sigma(p), \omega(p') < \omega(p), \text{ and } m_{p'} \in \text{im} \} \]

and let

\[ T_p = \{ p' \in P_\rightarrow \mid \tau(p') = \tau(p), \omega(p') < \omega(p), \text{ and } m_{p'} \in \text{im} \} \]

\textsc{Remark 6.36.} Note that \( T_p = \text{inv}(\Sigma_{\text{inv}(p)}) \).

\textsc{Remark 6.37.} Let \( p \) be a direction-preserving path of \( G \). For each path
\( p' \in \Sigma_p \cup T_p \), we have \( d(m_p, m_{p'}) = \omega(p) / 2 - \omega(p') / 2 \).

The set of undirected direction-preserving paths can be turned into
a graph by having an edge from an undirected direction-preserving

\[ \text{set } \Sigma_p \]

\[ \text{set } T_p \]
path to each such path that has one end in common with the path, has less weight than the path, and has the greatest weight among the paths with the former two properties. The edges can be weighted by the distance of the midpoints of the undirected paths. The graph and its edge-weighting are introduced in Definition 8.5:

\[ V_T = P_+ \leftrightarrow, \]

\[ E_T = \{(p_+ \leftrightarrow, p'_+ \leftrightarrow) \in V_T \times V_T \mid p' \in \arg\max_{p'' \in \Sigma_p \cup \Lambda_p} \omega(p'')\}, \]

and let

\[ \omega_T: E_T \to \mathbb{R}_{\geq 0}, \]

\[ ([p]_+, [p'_+]_+) \mapsto d(m_p, m_{p'}). \]

The triple \( G_T = (V_T, E_T, \omega_T) \) is an edge-weighted directed acyclic graph.

**Remark 8.39.** For each maximum-weight direction-preserving path \( \hat{p} \) in \( G \), the in-degree of \([\hat{p}]_+\) in \( G_T \) is 0, because there are only edges to equivalence classes of less-weight paths.

**Remark 6.40.** For each edge \( ([p]_+, [p'_+]_+) \) of \( G_T \), we have \( \omega_T([p]_+, [p'_+]_+) = \omega(p)/2 - \omega(p')/2 \).

**Remark 6.41.** There is a bijection between the vertices of \( V_T \) and the set of all midpoint signals that are created by the signal machine \( S \). The reason is that each midpoint signal memorises two words of directions in a set, one that leads from its position to one end of the path it designates the midpoint of and the other to the other end; because these words are stored in a set, the midpoint signal does not differentiate between source and target of its path.

Each maximum-weight vertex of \( G_T \) is a maximum-weight undirected direction-preserving path in \( G \) and is, under suitable identifications and definitions, a longest undirected direction-preserving path in the continuum representation \( M \) of \( G \). And, each minimum-weight vertex of \( G_T \) is one of weight 0, is an undirected empty path in \( G \), and is, under suitable identifications, an empty path in \( G \) and in \( M \), and a vertex in \( G \) and in \( M \).

For each path \( p_T \) in \( G_T \) that starts in a maximum-weight vertex and ends in a minimum-weight vertex, in the time evolution of the signal machine \( S \), there is a thaw signal that traverses the midpoint signals represented by the vertices on the path in the order they occur on the path such that the time the thaw signal takes to get from the vertex \( v_T \) on the path to the next vertex \( v'_T \) on the path is precisely the weight of the edge \( (v_T, v'_T) \), in particular, the time the thaw signal takes to reach the end of its path is the weight of \( p_T \).
To show that the synchronisation of all edges is thawed and finishes at the same time, we have to show that, for each edge, there is a thaw signal that collides with the midpoint signal of the edge and reaches the end of its path at one end of the edge, and that all thaw signals reach the ends of their paths at the same time. The former is equivalent to showing that, for each edge, there is a thaw signal that collides with the midpoint signal of the edge and reaches the end of its path at one end of the edge, and that all thaw signals reach the midpoint signals of all paths that have the same source or target as the edge, and therefore one from the midpoint signal of a path in $G_T$ from $[p]_{\leftrightarrow}$ to $[(v_0, e, v_1)]_{\leftrightarrow}$. And the latter is equivalent to showing that the weights of all paths from maximum-weight vertices to minimum-weight vertices in $G_T$ are equal to $d/2$. See corollary 6.46.

For each non-empty direction-preserving path $p$ in $G$, the midpoint of $p$ is found at time $\max\{d(g, \sigma(p)), d(g, \tau(p))\} + \omega(p)/2$ (see lemma 6.4) and, for each maximum-weight direction-preserving path $\hat{p}$ in $G$, the midpoint of $\hat{p}$ is found at time $r + d/2$ (see lemma 6.6). If the thaw signals that emanate from the midpoints of maximum-weight direction-preserving paths at the time $r + d/2$ reach the ends of their paths after the time span $d/2$, then synchronisation finishes at the optimal time $r + d$. The condition is equivalent to showing that the weights of all paths from maximum-weight vertices to minimum-weight vertices in $G_T$ are equal to $d/2$. See corollary 6.46.

The thaw signals that spread from the midpoint signal of a path $p$ eventually collide with the midpoint signals of all paths that have the same source or target as $p$, less weight, and whose midpoints lie on $p$. This is what is essentially shown in

**Lemma 6.42.** Let $p$ be a path in $G$ and let $p' \in \Sigma_p \cup T_p$. There is a path from $[p]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$ in $G_T$.

**Proof.** Case 1: $p' \in \Sigma_p$. If $p' \in \arg\max_{p'' \in \Sigma_p} \omega(p'')$, then $([p]_{\leftrightarrow}, [p']_{\leftrightarrow}) \in E_T$ and the path consisting of this edge is one from $[p]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$. Otherwise, there is a $q \in \arg\max_{p'' \in \Sigma_p} \omega(p'')$ such that $\omega(p') < \omega(q)$. Then, $([p]_{\leftrightarrow}, [q]_{\leftrightarrow}) \in E_T$. And, because $\{p', q\} \subseteq \Sigma_p$, we have $\sigma(p') = \sigma(p) = \sigma(q)$. Thus, because $\bar{q}((\omega(q)/2) = m_q \in \im \bar{p}$, the paths $q$ and $p$ are direction-preserving, and the multigraph $G$ is a tree, we have $\bar{q}([0, \omega(q)/2]) = \bar{p}([0, \omega(p)/2])$ and, analogously, we have $\bar{p}$ $[0, \omega(p')/2]$ $= \bar{q}([0, \omega(p')/2])$. Hence, because $\omega(p') < \omega(q)$,

$$m_{p'} = \bar{p}((\omega(p')/2) = \bar{p}(\omega(p')/2) = \bar{q}(\omega(p')/2) \in \im \bar{q}.$$

Therefore, $p' \in \Sigma_q$. Now, if $p' \in \arg\max_{p'' \in \Sigma_q} \omega(p'')$, then $([q]_{\leftrightarrow}, [p']_{\leftrightarrow}) \in E_T$, and the path consisting of the edges $([p]_{\leftrightarrow}, [q]_{\leftrightarrow})$ and $([q]_{\leftrightarrow}, [p']_{\leftrightarrow})$ is a path from $[p]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$. Otherwise, because $V_T$ is finite and $\omega(q) > \omega(p')$, it follows by induction that there is a path from $[q]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$, and therefore one from $[p]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$.
CASE 2: $p' \in T_p$. Then, $\text{inv}(p') \in \Sigma_{\text{inv}(p)}$. Hence, according to the first case, there is a path from $[\text{inv}(p)]_{\leftrightarrow}$ to $[\text{inv}(p')]_{\leftrightarrow}$. Therefore, because $[p]_{\leftrightarrow} = [\text{inv}(p)]_{\leftrightarrow}$ and $[p']_{\leftrightarrow} = [\text{inv}(p')]_{\leftrightarrow}$, there is a path from $[p]_{\leftrightarrow}$ to $[p']_{\leftrightarrow}$.

Each midpoint signal of a path eventually collides with a matching thaw signal that originated at the midpoint of a maximum-weight direction-preserving path. This is what is essentially shown in

LEMMA 6.43. Let $p$ be a direction-preserving path in $\mathcal{G}$. There is a maximum-weight direction-preserving path $\hat{p}$ in $\mathcal{G}$ such that there is a path from $[\hat{p}]_{\leftrightarrow}$ to $[p]_{\leftrightarrow}$ in $\mathcal{G}_T$. □

PROOF. If $p$ is a maximum-weight path, then the path $\hat{p} = p$ in $\mathcal{G}$ and the empty path $([p]_{\leftrightarrow})$ in $\mathcal{G}_T$ have the desired properties. From now on, let $p$ not be a maximum-weight path. Furthermore, let $\hat{p}$ be a maximum-weight path in $\mathcal{G}$ and let $p_d$ be the minimum-weight path in $\mathcal{G}$ such that $\sigma(p_d)$ lies on $p$ and $\tau(p_d)$ lies on $\hat{p}$. Then, there are paths $p_1$ and $p_2$ in $\mathcal{G}$ such that $p_1 \cdot p_2 = p$ and $\tau(p_1) = \sigma(p_d)$ as well as $\tau(\text{inv}(p_2)) = \sigma(p_d)$. And, there are paths $\hat{p}_1$ and $\hat{p}_2$ in $\mathcal{G}$ such that $\hat{p}_1 \cdot \hat{p}_2 = \hat{p}$ and $\tau(p_d) = \sigma(\hat{p}_2)$ as well as $\tau(p_d) = \sigma(\text{inv}(\hat{p}_1))$. Let

$$q_1 = \begin{cases} p_1, & \text{if } \omega(p_1) \geq \omega(p_2), \\ \text{inv}(p_2), & \text{otherwise}, \end{cases}$$

let

$$q_2 = \begin{cases} p_2, & \text{if } \omega(p_1) \geq \omega(p_2), \\ \text{inv}(p_1), & \text{otherwise}, \end{cases}$$

and let $q = q_1 \cdot q_2$. Furthermore, let

$$\hat{q}_1 = \begin{cases} \hat{p}_1, & \text{if } \omega(\hat{p}_2) \geq \omega(\hat{p}_1), \\ \text{inv}(\hat{p}_2), & \text{otherwise}, \end{cases}$$

let

$$\hat{q}_2 = \begin{cases} \hat{p}_2, & \text{if } \omega(\hat{p}_2) \geq \omega(\hat{p}_1), \\ \text{inv}(\hat{p}_1), & \text{otherwise}, \end{cases}$$

and let $\hat{q} = \hat{q}_1 \cdot \hat{q}_2$. Moreover, let $p' = q_1 \cdot p_d \cdot \hat{q}_2$. Then, $q \in \{p, \text{inv}(p)\}$ as well as $\hat{q} \in \{\hat{p}, \text{inv}(\hat{p})\}$. And, $\omega(q_1) \geq \omega(q_2)$ as well as $\omega(\hat{q}_1) \leq \omega(\hat{q}_2)$. And, $\tau(p') = \tau(\hat{q})$ as well as $\sigma(q) = \sigma(p')$. And, because $\omega(q_1) \geq \omega(q_2)$, we have $m_q \in \text{im}(\hat{q}_1) \subseteq \text{im}p'$. 

CASE 1: $p'$ is a maximum-weight path. Then, because $p$ is not a maximum-weight path, we have $\omega(q) = \omega(p) < \omega(p')$. Hence, because $\sigma(q) = \sigma(p')$ and $m_q \in \text{im}p'$, we have $q \in \Sigma_{p'}$. In conclusion, according to lemma 6.42, there is a path from $[p']_{\leftrightarrow}$ to $[p]_{\leftrightarrow} = [q]_{\leftrightarrow}$.
CASE 2: \( p' \) is not a maximum-weight path. Then, \( \omega(q_1 \bullet p_d \bullet \hat{q}_2) = \omega(p') < \omega(\hat{q}) = \omega(\hat{q}_1 \bullet \hat{q}_2) \) and thus \( \omega(q_1 \bullet p_d) < \omega(\hat{q}_1) \), in particular, \( \omega(q_1) < \omega(\hat{q}_1) \). Hence, because \( \omega(q_2) \leq \omega(q_1) < \omega(\hat{q}_1) \leq \omega(\hat{q}_2) \), we have \( \omega(q) = \omega(q_1) + \omega(q_2) < \omega(q_1) + \omega(\hat{q}_2) \leq \omega(p') \).

And, because \( \omega(q_1 \bullet q_d) < \omega(\hat{q}_1) \leq \omega(\hat{q}_2) \), we have \( m_{p'} \in \text{im}\hat{q}_2 \subseteq \text{im}\hat{q} \). And, recall that \( m_q \in \text{im}\hat{p} \). Altogether, because \( \tau(p') = \tau(\hat{q}) \) and \( \sigma(q) = \sigma(p') \), we have \( p' \in T_\hat{q} \) and \( q \in \Sigma_{p'} \). Therefore, according to lemma 6.42, there is a path from \([\hat{p}]{\leftrightarrow} \) to \([p']\) and there is a path from \([p']\to [q] \) to \([p] \). In conclusion, there is a path from \([\hat{p}]{\leftrightarrow} \) to \([p]\). □

The time it takes a thaw signal from the midpoint of a path to collide with the midpoint signal of a matching path is given by half the former path’s length minus half the latter path’s length. This is what is essentially shown in

**Lemma 6.44.** For each path \( p_T \) in \( G_T \), the weight of \( p_T \) is equal to \( \omega(\sigma(p_T))/2 - \omega(\tau(p_T))/2 \).

□

**Proof.** We prove the statement by induction.

**Base Case.** Each empty path \( p_T \) in \( G_T \) has weight 0, has the same source and target vertices, and \( \omega(\sigma(p_T))/2 - \omega(\tau(p_T))/2 \) is equal to 0 as needed.

**Inductive Step.** Let \( p_T = ([p_0]_{\leftrightarrow}, [p_1]_{\leftrightarrow}, \ldots, [p_n]_{\leftrightarrow}) \) be a non-empty path in \( G_T \) such that the weight of the subpath \(([p_1]_{\leftrightarrow}, \ldots, [p_n]_{\leftrightarrow}) \) is equal to \( \omega([p_1]_{\leftrightarrow})/2 - \omega([p_n]_{\leftrightarrow})/2 \). Then, according to remark 6.40, we have \( \omega_T([p_0]_{\leftrightarrow}, [p_1]_{\leftrightarrow}) = d(m_{p_0}, m_{p_1}) = \omega(p_0)/2 - \omega(p_1)/2 = \omega([p_0]_{\leftrightarrow})/2 - \omega([p_1]_{\leftrightarrow})/2 \). Hence, the weight of the path \( p_T \) is equal to \( \omega([p_0]_{\leftrightarrow})/2 - \omega([p_1]_{\leftrightarrow})/2 + \omega([p_1]_{\leftrightarrow})/2 - \omega([p_0]_{\leftrightarrow})/2 = \omega([p_0]_{\leftrightarrow})/2 - \omega([p_n]_{\leftrightarrow})/2 \).

□

The time it takes a thaw signal from a maximum-weight direction-preserving path to collide with the midpoint signal of a matching path is essentially given in

**Corollary 6.45.** For each maximum-weight direction-preserving path \( \hat{p} \) in \( G \) such that there is a path from \([\hat{p}]_{\leftrightarrow} \) to \([p]_{\leftrightarrow} \) in \( G_T \), the weight of this path is \( d/2 - \omega(p)/2 \).

□

**Proof.** This is a direct consequence of lemma 6.44 with the fact that \( \omega(\hat{p}) = d \).

□

The time it takes a thaw signal from a maximum-weight direction-preserving path to collide with the midpoint signal of an edge it thaws and to reach the ends of the edge is essentially given in
Corollary 6.46. Let $e$ be an edge of $G$, and let $v_0$ and $v_1$ be the two ends of $e$. There is a maximum-weight direction-preserving path $\hat{p}$ such that there is a path from $[\hat{p}]_{e}$ to $[\langle v_0, e, v_1 \rangle]_{e}$, and all such paths have weight $d/2 - \omega(e)/2$, and there are also paths from $[\hat{p}]_{e}$ to $[\langle v_0 \rangle]_{e}$ and to $[\langle v_1 \rangle]_{e}$, and all such paths have weight $d/2$. □

Proof. This is a direct consequence of lemma 6.43 and corollary 6.45.

References

[Bal67] Robert Balzer. ‘An 8-state minimal time solution to the firing squad synchronization problem’. In: Information and Control 10.1 (1967), pages 22–42 (cited on page 2).

[CD91] Karel Culik II and Simant Dube. ‘An efficient solution of the firing mob problem’. In: Theoretical Computer Science 91.1 (1991), pages 57–69 (cited on page 2).

[Duro5] Jérôme Olivier Durand-Lose. ‘Abstract Geometrical Computation: Turing-Computing Ability and Undecidability’. In: Lecture Notes in Computer Science 3526 (2005), pages 106–116 (cited on page 3).

[Duro8] Jérôme Olivier Durand-Lose. ‘The signal point of view: from cellular automata to signal machines’. In: Proceedings of the First Symposium on Cellular Automata "Journées Automates Cellulaires" (JAC 2008), Uzès, France, April 21-25, 2008. Edited by Bruno Durand. MCCME Publishing House, Moscow, 2008, pages 238–249 (cited on page 29).

[Ger87] Hein D. Gerken. ‘Über Synchronisationsprobleme bei Zellularamaten’. German. Diplomarbeit. Braunschweig: Technische Universität Braunschweig, 1987 (cited on page 2).

[Got62] Eiichi Goto. ‘A minimal time solution of the firing squad problem’. In: Dittoed course notes for Applied Mathematics. Harvard University 298 (1962), pages 52–59 (cited on page 2).

[Gre83] John J. Grefenstette. ‘Network Structure and the Firing Squad Synchronization problem’. In: Computer and System Sciences 26.1 (1983), pages 139–152 (cited on pages 2, 3).

[Kob77] Kojiro Kobayashi. ‘The firing squad synchronization problem for two-dimensional arrays’. In: Information and Control 34.3 (1977), pages 177–197 (cited on page 2).
[Kob78a] Kojiro Kobayashi. ‘On the minimal firing time of the firing squad synchronization problem for polyautomata networks’. In: *Theoretical Computer Science* 7.2 (1978), pages 149–167 (cited on page 2).

[Kob78b] Kojiro Kobayashi. ‘The firing squad synchronization problem for a class of polyautomata networks’. In: *Computer and System Sciences* 17.3 (1978), pages 300–318 (cited on page 2).

[Maz87] Jacques Mazoyer. ‘A Six-state Minimal Time Solution to the Firing Squad Synchronization Problem’. In: *Theoretical Computer Science* 50.2 (1987), pages 183–238 (cited on pages 2, 3, 60).

[Moo64] Edward Forrest Moore. *Sequential machines. Selected papers*. Computer Science and Information Processing. Addison-Wesley, 1964, pages 213–214 (cited on page 1).

[RFH72] P. Rosenstiehl, J. R. Fiksel and A. Holliger. ‘Intelligent graphs. Networks of finite automata capable of solving graph problems’. In: *Graph Theory and Computing*. Edited by Ronald C. Read. Academic Press, 1972, pages 219–265 (cited on page 2).

[Rók00] Zsuzsanna Róka. ‘The firing squad synchronization problem on Cayley graphs’. In: *Theoretical Computer Science* 244.1 (2000), pages 243–256 (cited on page 2).

[Rom76] Francesco Romani. ‘Cellular automata synchronization’. In: *Information Sciences* 10.2 (1976), pages 299–318 (cited on page 2).

[San94] Peter Sanders. ‘Massively Parallel Search for Transition-Tables of Polyautomata’. In: *Parcella ’94*. Proceedings of the Vi. International Workshop on Parallel Processing by Cellular Automata and Arrays. Volume 81. Mathematical Research. Potsdam, Germany: Akademie, 1994, pages 99–108 (cited on page 2).

[UYY09] Hiroshi Umeo, Jean-Baptiste Yunes and Takuya Yamawaki. ‘A Simple Optimum-Time Firing Squad Synchronization Algorithm for Two-Dimensional Rectangle Arrays’. In: *CSSim ’09*. Proceedings of the First International Conference on Computational Intelligence, Modelling and Simulation. Brno, TBD, Czech Republic: IEEE, 2009, pages 120–125 (cited on page 2).

[Wac16] Simon Wacker. ‘Cellular Automata on Group Sets and the Uniform Curtis-Hedlund-Lyndon Theorem’. In: *Cellular Automata and Discrete Complex Systems*. Proceedings of the 22nd IFIP WG 1.5 International Workshop, AUTOMATA
Simon Wacker. *Right Amenability And Growth Of Finitely Right Generated Left Group Sets*. 2017. arXiv: 1701.02108 [math.GR]. Unpublished (cited on page 4).

Abraham Waksman. ‘An optimum solution to the firing squad synchronization problem’. In: *Information and Control* 9.1 (1966), pages 66–78 (cited on page 2).