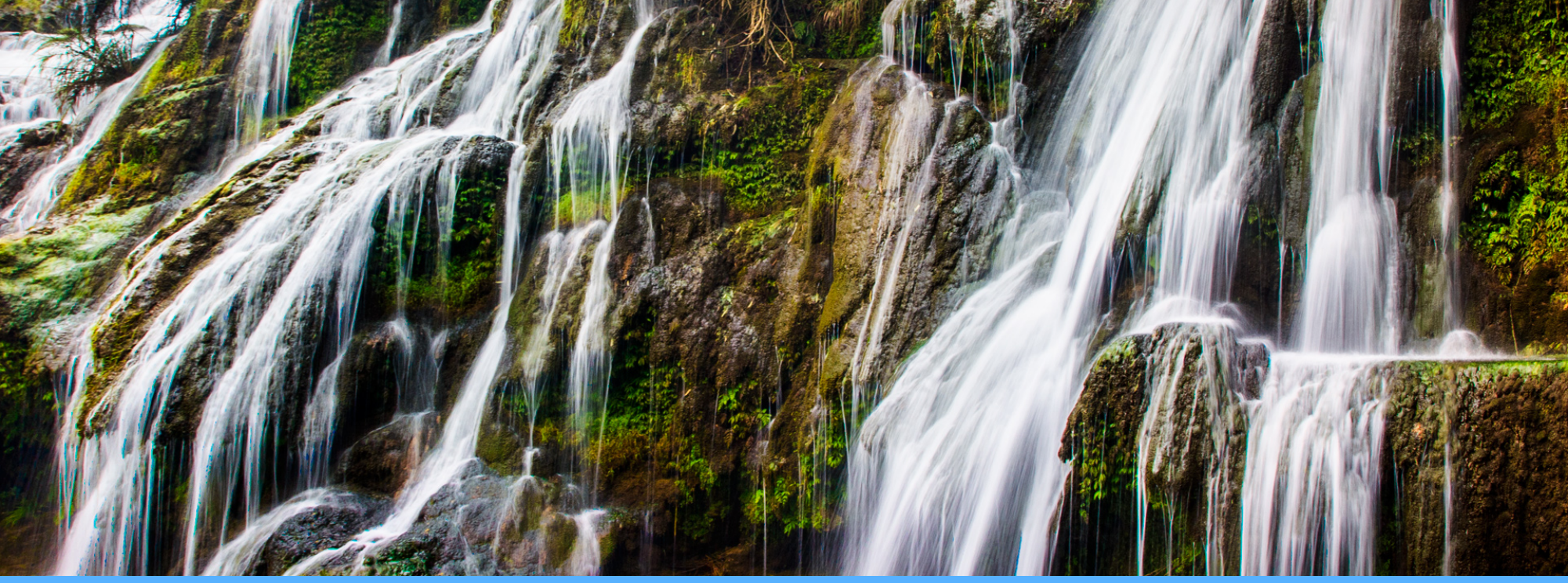


Degrees of randomized computability

Rupert Hölzl
Bundeswehr University Munich

Joint work with Christopher P. Porter
Drake University, Des Moines

1 Motivation & Preliminaries

Paradigm

"What can be computed with access to randomness?"

Sacks' theorem vs. collections of sequences

1. **Theorem (Sacks).** If A is computable from every X in a set of positive measure, then A is computable.

Sacks' theorem vs. collections of sequences

1. **Theorem (Sacks).** If A is computable from every X in a set of positive measure, then A is computable.

2. But the situation looks very different for *collections* of sets, as almost all oracles compute
   - a set of hyperimmune degree,
   - a 1-generic set,
   - a Martin-Löf random set.

Randomized computation of elements of collections

1. **Definition.** Let $(\Phi_i)_{i\in\omega}$ be an effective enumeration of all Turing functionals. Then a measurable collection $\mathscr{A} \subseteq 2^\omega$ is called *negligible* if and only if

$$\lambda\left(\bigcup_{i\in\omega} \Phi_i^{-1}(\mathscr{A})\right) = 0.$$

2. So the *non*negligible collections are those for which there exists (at least one) algorithm that can produce an element of the collection with positive probability.
3. This model of randomized computation is the topic of this talk.

Semimeasures and their atoms

1. **Definition.** A *semimeasure* is a function $P\colon 2^{<\omega} \to [0,1]$ with
   - $P(\varepsilon) = 1$,
   - $P(\sigma) \geq P(\sigma 0) + P(\sigma 1)$ for every $\sigma \in 2^{<\omega}$.

   In addition, P is *left-c.e.* if $P(\sigma)$ is the limit of a computable nondecreasing sequences of rationals, uniformly in $\sigma \in 2^{<\omega}$.
2. **Definition.** The *support* of a semimeasure P is

$$\mathrm{Supp}(P) := \{X \in 2^{\omega} : \forall n\, P(X\restriction n) > 0\}.$$

3. **Definition.** $X \in 2^{\omega}$ is an *atom* of P if there is some $\delta > 0$ such that $P(X\restriction n) > \delta$ for all n.

Semimeasures and measures

1. **Definition.** Given a semimeasure P and $\sigma \in 2^{<\omega}$ we define

$$\overline{P}(\sigma) = \inf_n \sum_{\sigma \preceq \tau \& |\tau| = n} P(\tau).$$

   $\overline{P}$ induces a measure on 2^ω by letting $\overline{P}([\![\sigma]\!]) := \overline{P}(\sigma)$.

2. **Proposition.** For any left-c.e. semimeasure P and any $\mathscr{A} \subseteq 2^\omega$, if $\overline{P}(\mathscr{A}) > 0$, then $\mathscr{A}$ is nonnegligible.

2 The Levin-V'yugin algebra

The LV-reduction

1. **Recall:** $\mathscr{A} \subseteq 2^\omega$ is *Turing invariant* if

$$X \in \mathscr{A} \wedge Y \equiv_{\mathrm{T}} X \Longrightarrow Y \in \mathscr{A}.$$

2. Let $\mathscr{I}$ denote the measurable Turing invariant subsets of 2^ω.
3. **Definition.** Let $\mathscr{A}, \mathscr{B} \in \mathscr{I}$.
   - $\mathscr{A} \leq_{\mathrm{LV}} \mathscr{B}$ if and only if $\mathscr{A} \setminus \mathscr{B}$ is negligible.
   - $\mathscr{A} \equiv_{\mathrm{LV}} \mathscr{B}$ if and only if $\mathscr{A} \leq_{\mathrm{LV}} \mathscr{B}$ and $\mathscr{B} \leq_{\mathrm{LV}} \mathscr{A}$.
4. **Intuition.** $\mathscr{A} \leq_{\mathrm{LV}} \mathscr{B}$ says that, for any probabilistic algorithm, the probability that it produces an element of $\mathscr{A} \setminus \mathscr{B}$, is 0.
5. **Intuition.** $\mathscr{A} <_{\mathrm{LV}} \mathscr{B}$ says in addition that, for some probabilistic algorithm, the probability that it produces an element of $\mathscr{B}$ that is not in $\mathscr{A}$, is strictly positive.
6. **Intuition.** The larger a collection of sets is in the given order, the easier it is to probabilistically produce an element of it.

The Levin-V'yugin algebra

1. **Definition.** We call $\mathscr{D}_{\mathrm{LV}} = \mathscr{I}/\equiv_{\mathrm{LV}}$ the *Levin-V'yugin algebra*. Elements of $\mathscr{D}_{\mathrm{LV}}$ will be referred to as *LV-degrees*.
2. **Notation.** Let $\mathbf{deg_{LV}}(\mathscr{A})$ denote the LV-degree of $\mathscr{A} \in \mathscr{I}$.
3. **Definition.** Given $\mathscr{A} \in \mathbf{a} \in \mathscr{D}_{\mathrm{LV}}$ and $\mathscr{B} \in \mathbf{b} \in \mathscr{D}_{\mathrm{LV}}$, define
   - $\mathbf{a} \wedge \mathbf{b} := \mathbf{deg_{LV}}(\mathscr{A} \cap \mathscr{B})$,
   - $\mathbf{a} \vee \mathbf{b} := \mathbf{deg_{LV}}(\mathscr{A} \cup \mathscr{B})$, and
   - $\mathbf{a}^c = \mathbf{deg_{LV}}(2^\omega \setminus \mathscr{A})$.

   It is easy to see that these are welldefined.
4. **Proposition.** The bottom element $\mathbf{0}$ of $\mathscr{D}_{\mathrm{LV}}$ consists of the Turing invariant negligible subsets of 2^ω.
5. **Proposition.** The top element $\mathbf{1}$ of $\mathscr{D}_{\mathrm{LV}}$ consists of all Turing invariant $\mathscr{A} \subseteq 2^\omega$ such that $2^\omega \setminus \mathscr{A}$ is negligible.

Atoms of the algebra

1. **Definition.** $\mathbf{a} \in \mathscr{D}_{\mathrm{LV}}$ is an *atom* if there are no $\mathscr{D}_{\mathrm{LV}} \ni \mathbf{b}, \mathbf{c} \neq 0$ such that $\mathbf{a} = \mathbf{b} \vee \mathbf{c}$ and $\mathbf{b} \wedge \mathbf{c} = 0$.

   (To avoid confusion with the atoms of a semimeasure, we will talk about $\mathscr{D}_{\mathrm{LV}}$*-atoms*.)
2. **Proposition (V'yugin).** $\mathbf{c} := \mathbf{deg_{LV}}(\mathrm{REC})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.
3. **Theorem (V'yugin).** $\mathbf{r} := \mathbf{deg_{LV}}((\mathrm{MLR})^{\equiv_{\mathrm{T}}})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.

Atoms of the algebra

1. **Definition.** $\mathbf{a} \in \mathscr{D}_{\mathrm{LV}}$ is an *atom* if there are no $\mathscr{D}_{\mathrm{LV}} \ni \mathbf{b}, \mathbf{c} \neq 0$ such that $\mathbf{a} = \mathbf{b} \vee \mathbf{c}$ and $\mathbf{b} \wedge \mathbf{c} = 0$.

   (To avoid confusion with the atoms of a semimeasure, we will talk about $\mathscr{D}_{\mathrm{LV}}$-*atoms*.)
2. **Proposition (V'yugin).** $\mathbf{c} := \mathbf{deg_{LV}}(\mathrm{REC})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.
3. **Theorem (V'yugin).** $\mathbf{r} := \mathbf{deg_{LV}}((\mathrm{MLR})^{\equiv_{\mathrm{T}}})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.
4. **Theorem (V'yugin).** $\mathbf{r} \vee \mathbf{c} \neq 1$.

   We work towards the proof in a moment.

Atoms of the algebra

1. **Definition.** $\mathbf{a} \in \mathscr{D}_{\mathrm{LV}}$ is an *atom* if there are no $\mathscr{D}_{\mathrm{LV}} \ni \mathbf{b}, \mathbf{c} \neq 0$ such that $\mathbf{a} = \mathbf{b} \vee \mathbf{c}$ and $\mathbf{b} \wedge \mathbf{c} = 0$.

   (To avoid confusion with the atoms of a semimeasure, we will talk about $\mathscr{D}_{\mathrm{LV}}$-*atoms*.)

2. **Proposition (V'yugin).** $\mathbf{c} := \mathbf{deg_{LV}}(\mathrm{REC})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.

3. **Theorem (V'yugin).** $\mathbf{r} := \mathbf{deg_{LV}}((\mathrm{MLR})^{\equiv_{\mathrm{T}}})$ is a $\mathscr{D}_{\mathrm{LV}}$-atom.

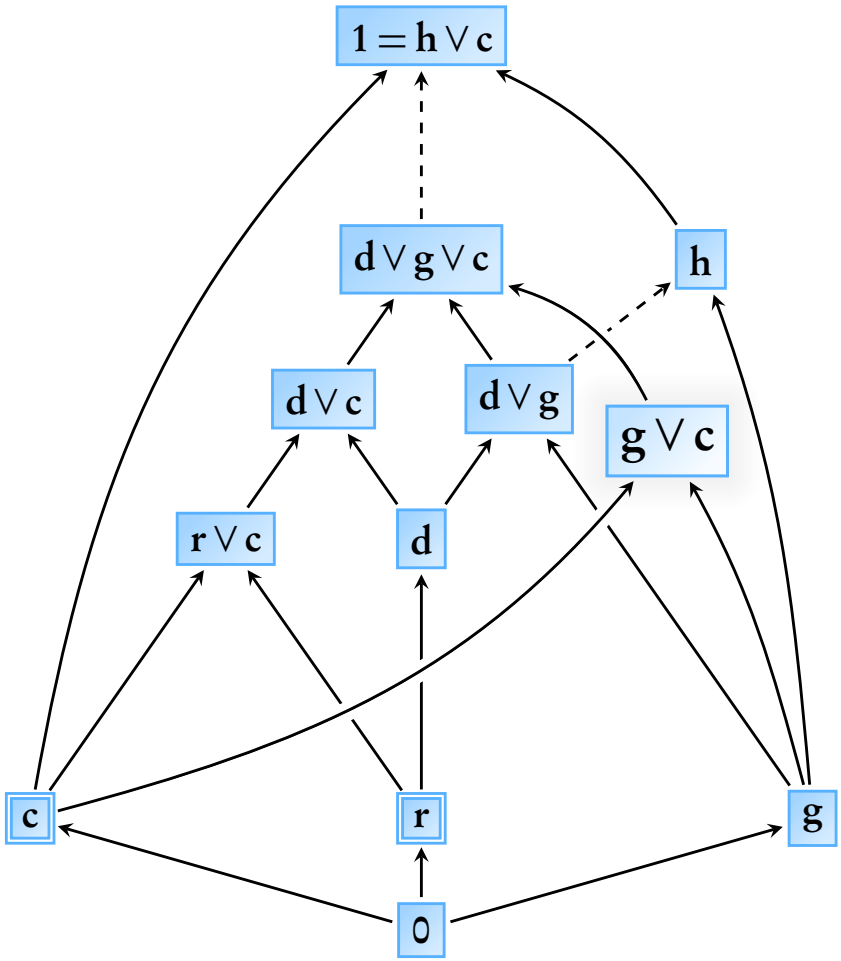
4. **Theorem (V'yugin).** $\mathbf{r} \vee \mathbf{c} \neq 1$.

   We work towards the proof in a moment.

5. **Question.** How many atoms are there?

   We discuss this in the last part of the talk.

Known structure of the algebra



1. Standard arrows represent strict separations in the LV-degrees.
2. **Question.** Is $\mathbf{g}$ a $\mathscr{D}_{\mathrm{LV}}$-atom?
3. **Question.** Is $\mathbf{d} \vee \mathbf{g} = \mathbf{h}$, and thus is $\mathbf{d} \vee \mathbf{g} \vee \mathbf{c} = \mathbf{1}$?

Semimeasures as flows

A first goal

1. **Theorem (V'yugin).** There is a left-c.e. semimeasure P with
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X) \notin \mathrm{MLR}$.
2. **Corollary.** $\mathbf{r} \vee \mathbf{c} \neq \mathbf{1}$.

   Proof. Write $\mathscr{A}$ for $\mathrm{Supp}(P) \setminus \mathrm{REC}$. Since P has no atoms, $\overline{P}(\mathscr{A}) = \overline{P}(\mathrm{Supp}(P)) > 0$ and so $\mathscr{A}$ is nonnegligible.

   But clearly, $\mathbf{deg_{LV}}((\mathscr{A})^{\equiv_T}) \wedge (\mathbf{r} \vee \mathbf{c}) = \mathbf{0}$. □

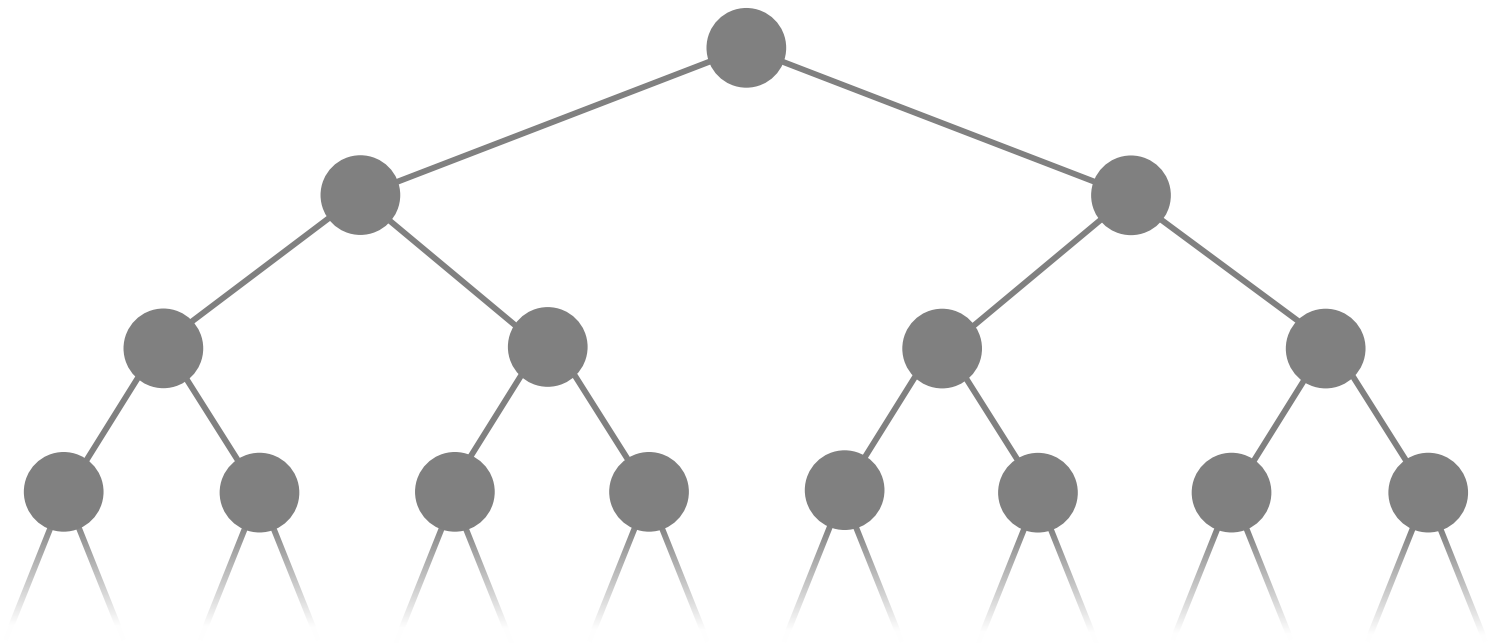
A first goal

1. **Theorem (V'yugin).** There is a left-c.e. semimeasure P with
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X) \notin \mathrm{MLR}$.
2. **Corollary.** $\mathbf{r} \vee \mathbf{c} \neq \mathbf{1}$.

   Proof. Write $\mathscr{A}$ for $\mathrm{Supp}(P) \setminus \mathrm{REC}$. Since P has no atoms, $\overline{P}(\mathscr{A}) = \overline{P}(\mathrm{Supp}(P)) > 0$ and so $\mathscr{A}$ is nonnegligible.

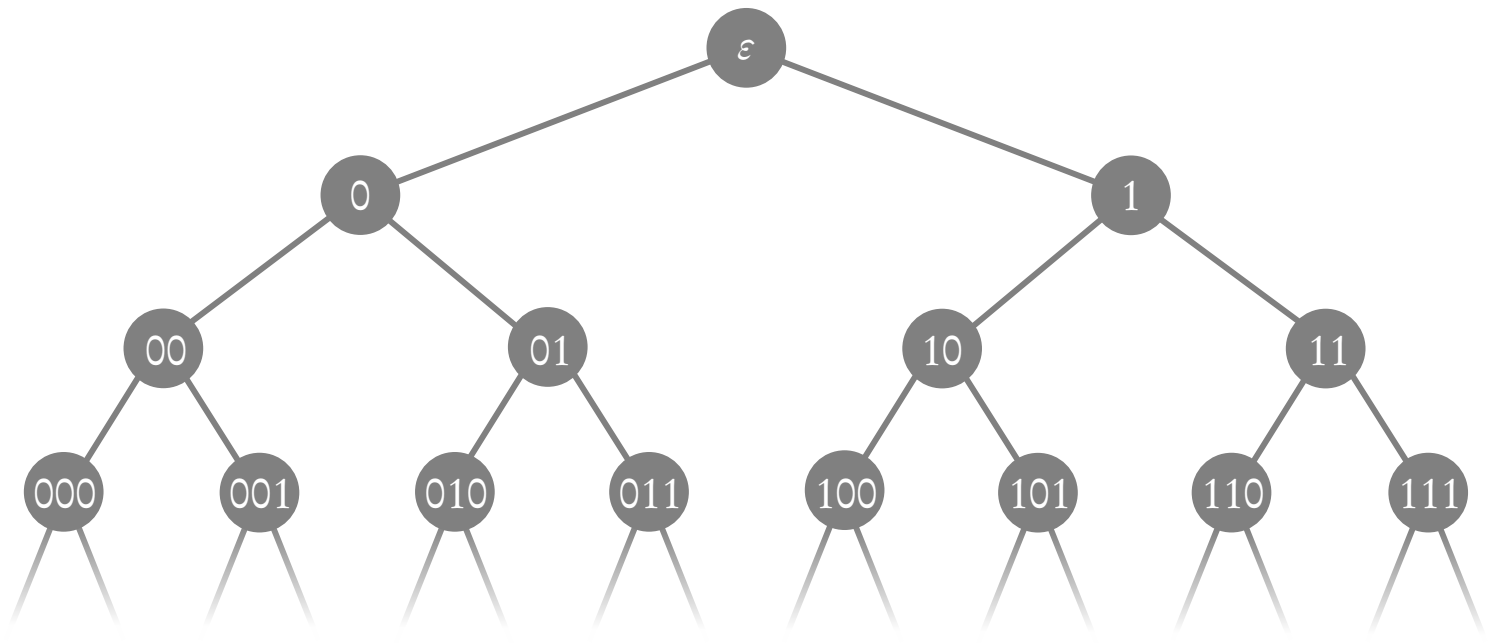
   But clearly, $\mathbf{deg_{LV}}((\mathscr{A})^{\equiv_T}) \wedge (\mathbf{r} \vee \mathbf{c}) = \mathbf{0}$. □
3. To prove the theorem, we need to build a semimeasure with certain computational properties. V'yugin has a special finite injury technique for that, which we study now.

Flows on Cantor space



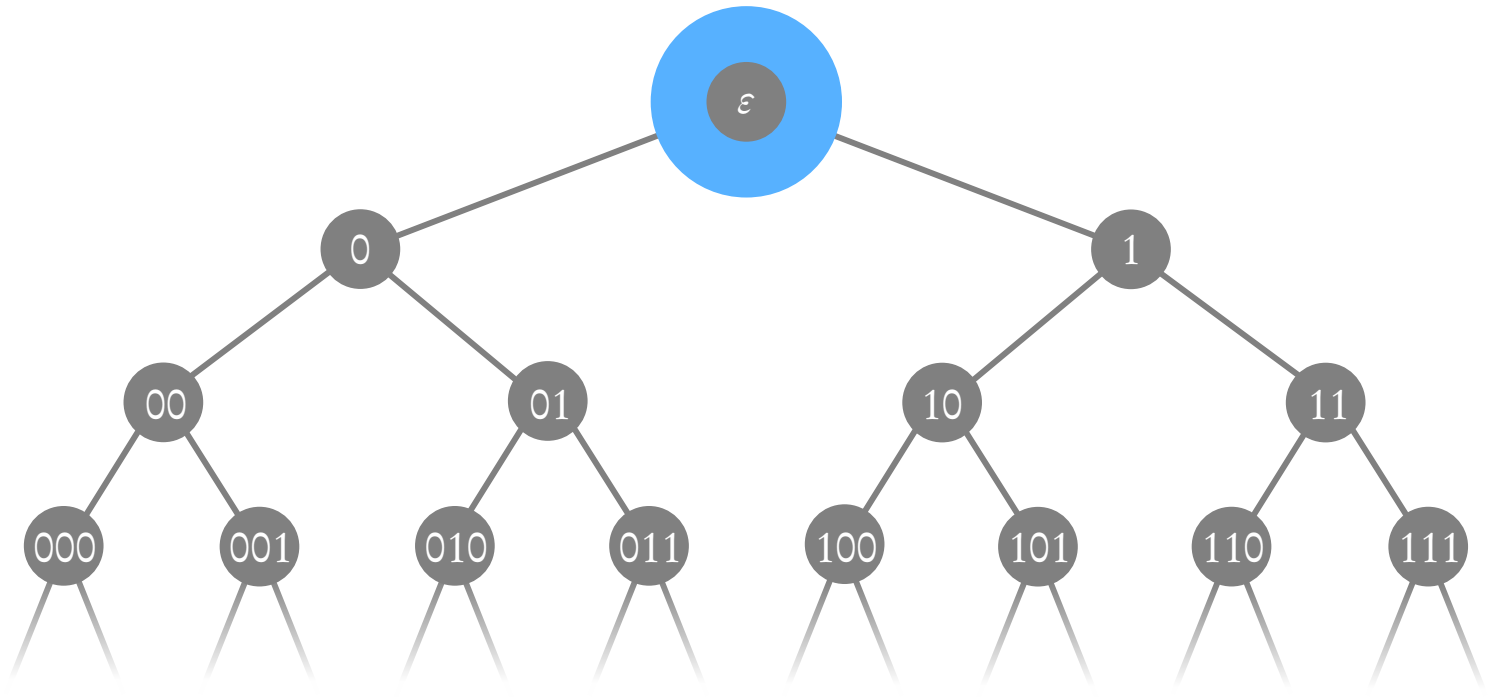
We look at the binary tree.
The gray, downward directed edges are called *normal edges*.

Flows on Cantor space



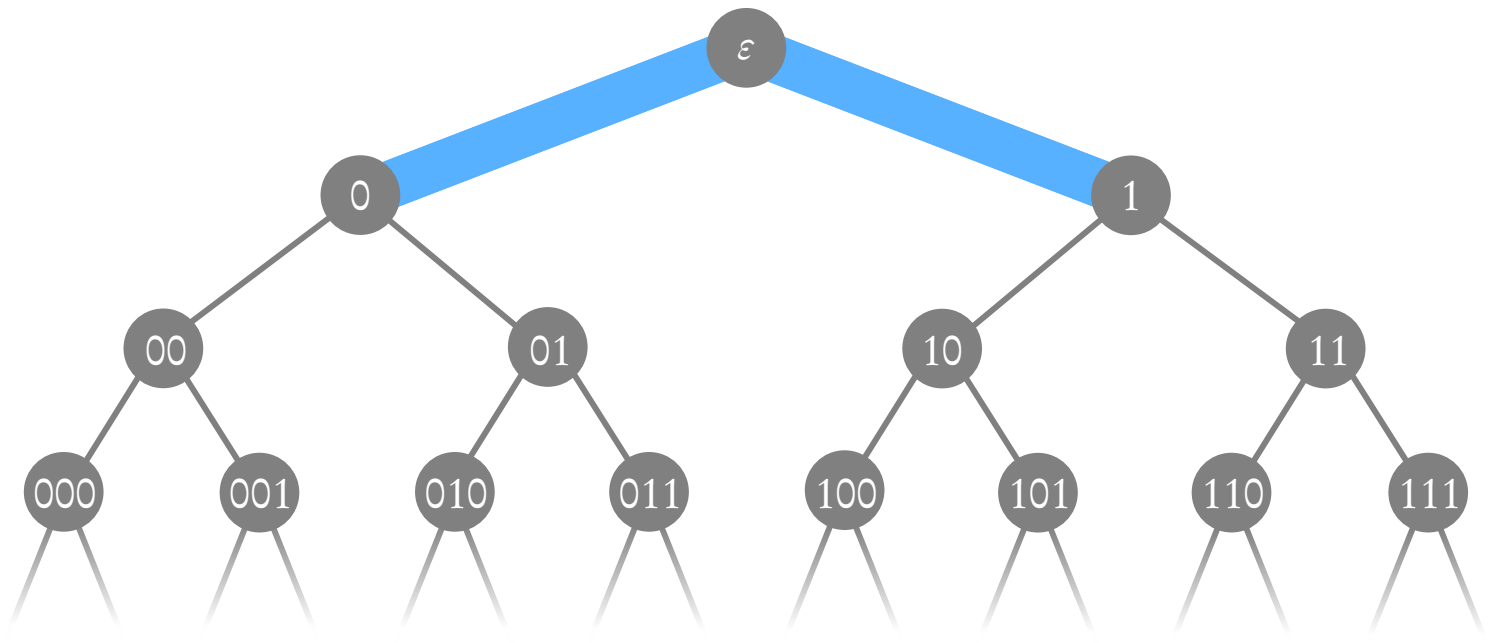
We identify the nodes in the tree with the finite binary strings.

Flows on Cantor space



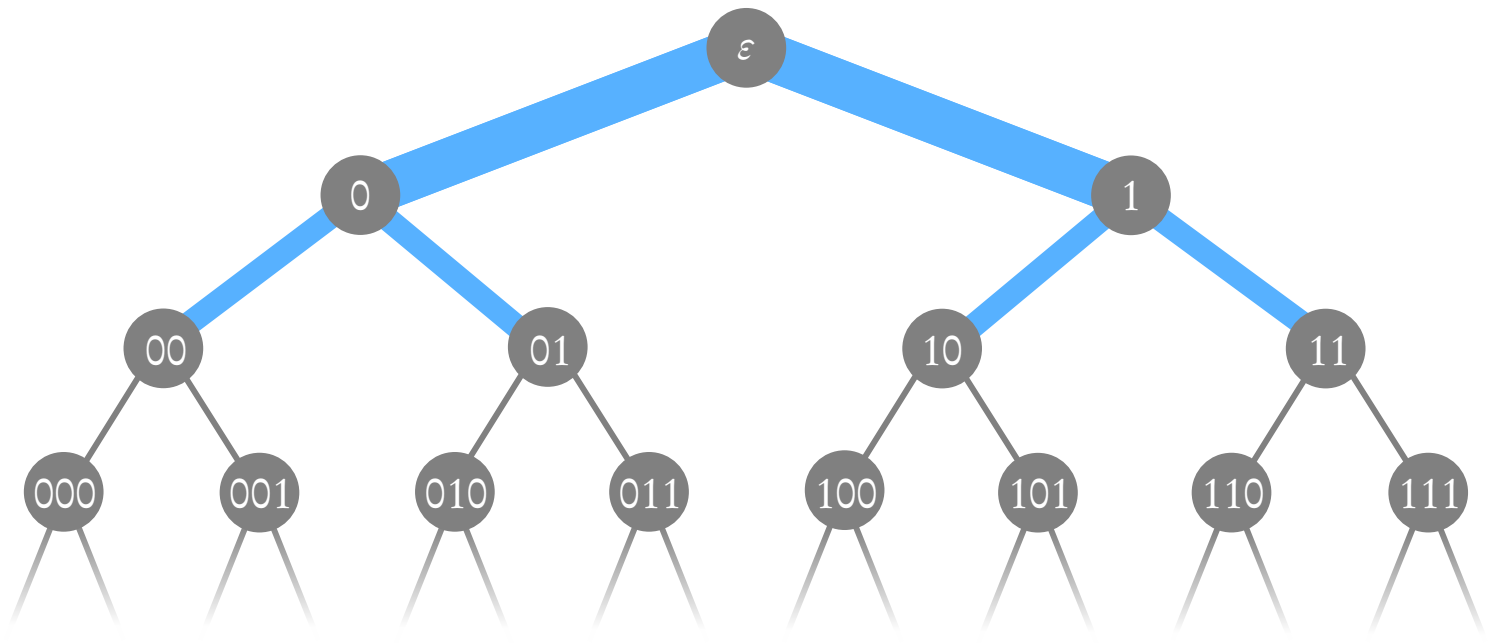
We put amount 1 of flow at the root ε.

Flows on Cantor space

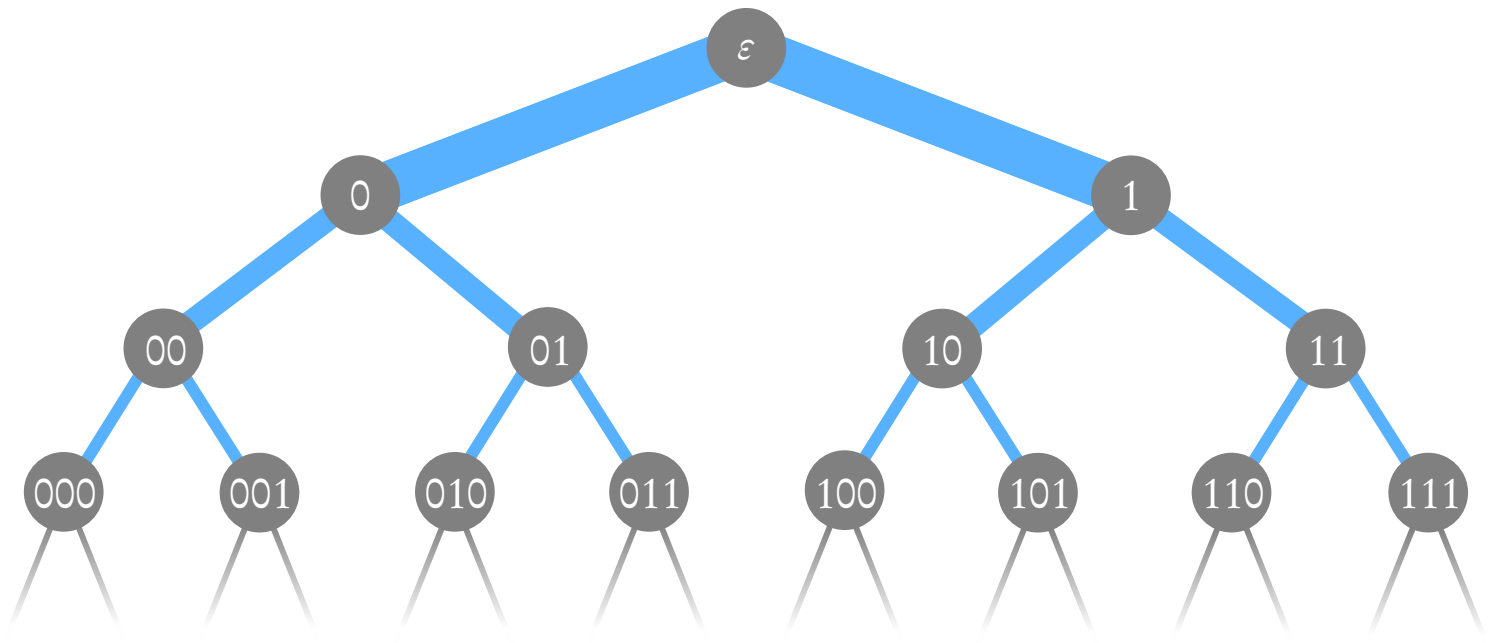


Evenly split, this measure flows to the two children of ε …

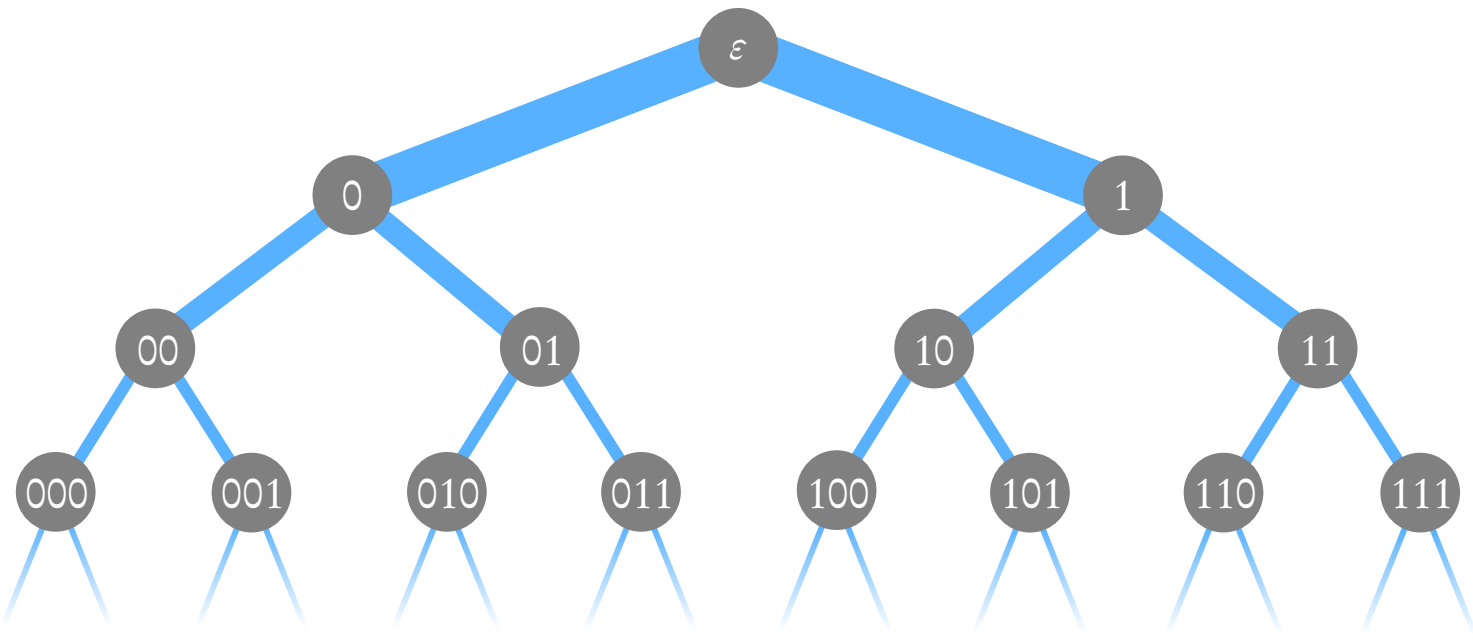
Flows on Cantor space



Flows on Cantor space

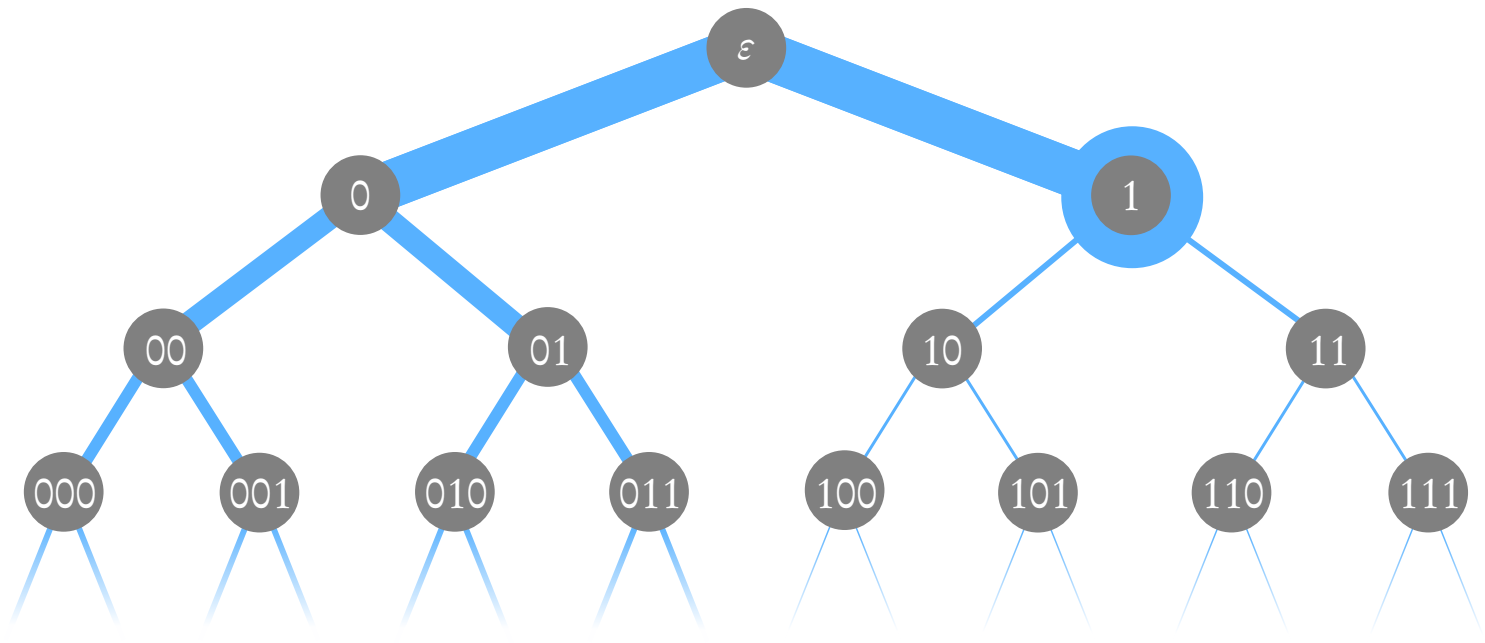


Flows on Cantor space



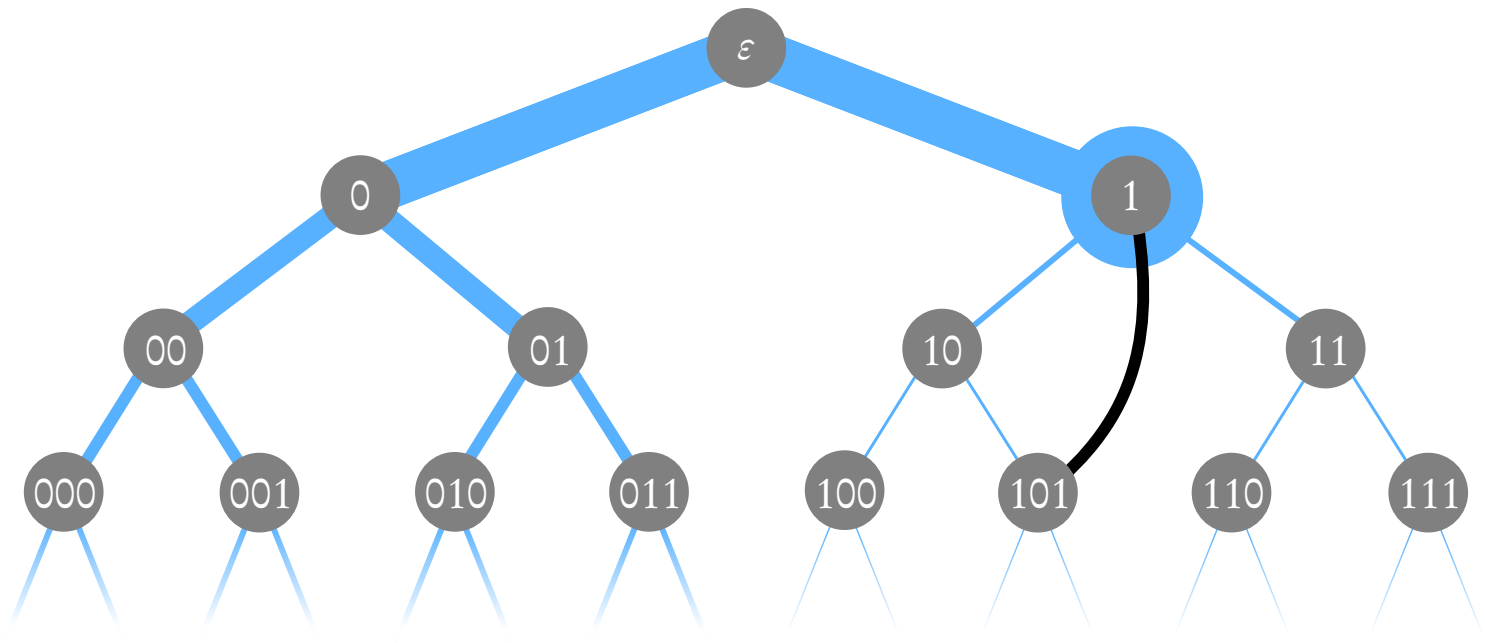
…and so on. Clearly, this induces the Lebesgue measure.

Flows on Cantor space



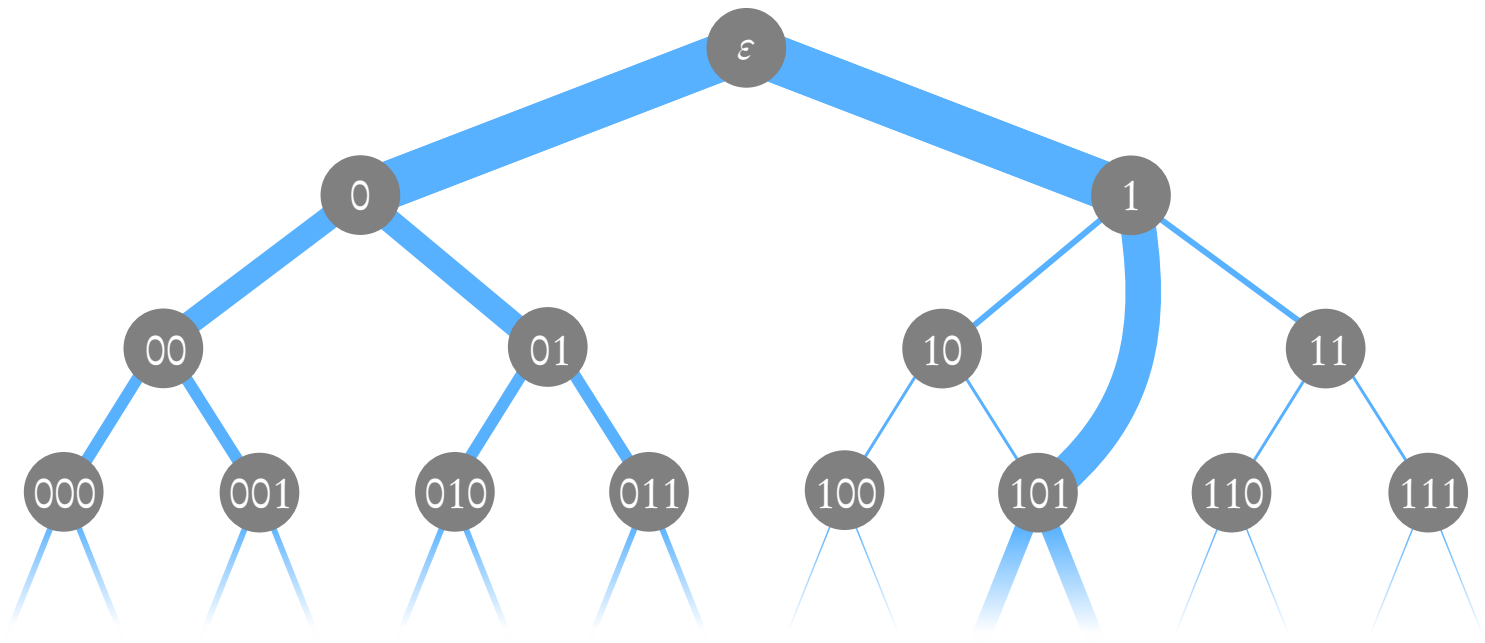
But now we allow holding back flow at nodes, forming a puddle.
Less flow arrives at the nodes' children consequently.

Flows on Cantor space



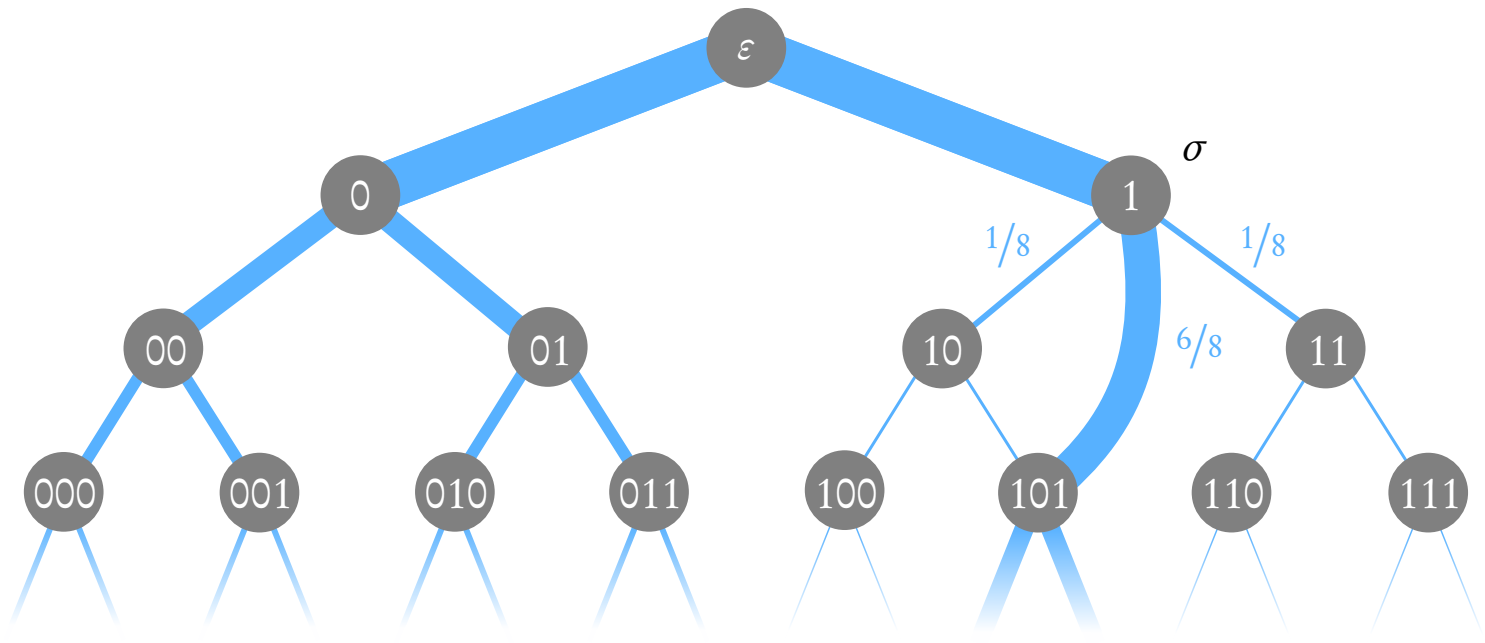
At some later stage we *may* decide to add a new *extra edge* to the graph.

Flows on Cantor space



The previously held back flow can now be pushed through that new edge, again affecting the children's flow accordingly.

Flows on Cantor space



We denote by $q(\sigma, \tau)$ the *fraction* of the flow arriving at σ that gets pushed through the edge going from σ to τ.

From flow to semimeasure

1. **Definition.** The *amount of flow into a node* τ, is defined via

$$R(\varepsilon) = 1,$$
$$R(\tau) = \sum_{(\sigma,\tau)\in\mathscr{E}} q(\sigma,\tau)R(\sigma).$$

where $\mathscr{E}$ is the collection of *all* edges.

From flow to semimeasure

1. **Definition.** The *amount of flow into a node* τ, is defined via

$$R(\varepsilon) = 1,$$
$$R(\tau) = \sum_{(\sigma,\tau)\in\mathscr{E}} q(\sigma,\tau)R(\sigma).$$

   where $\mathscr{E}$ is the collection of *all* edges.
2. $\sigma \prec \tau$ does not imply $R(\sigma) \geq R(\tau)$ as not all of the flow observed at τ must have flown through σ. There could be an extra edge bypassing σ and diverting flow around it.
3. In particular, R is not a semimeasure, so we need to correct for this lack of monotonicity.

From flow to semimeasure

1. Let T_σ be all finite prefix-free sets of strings τ such that $\sigma \preceq \tau$.
2. **Definition.** Let R be an inflow function associated with some q. Then the *q-flow P* is defined by

$$P(\sigma) = \sup_{D \in T_\sigma} \sum_{\tau \in D} R(\tau).$$

3. **Intuition.** $P(\sigma)$ is the largest amount of flow that can be observed passing through a set of extensions of the node σ.

   The motivation for looking at prefixfree sets D of nodes is to avoid counting the same quantity of flow more than once.
4. **Lemma.** If q is computable, then P is a left-c.e. semimeasure.

Objective of the construction

1. **Goal:** All sequences in the support of the constructed semimeasure have certain computability-theoretic properties.
2. So those sequences who do not have those properties need to be removed from the support after finitely many steps.
3. **Problem:** It is only eventually recognizable that a sequence has one of the properties; we may remove a sequence prematurely.

Approach

1. Each sequence gets multiple chances to show it has the property.
2. If a sequences turns out to have the property, we "put it in the bank" by pushing a lot of flow into it. To have that flow available, we have previously held it back at a parent node.
3. If after multiple attempts a sequence still has not demonstrated that it has the desired property, we remove the sequence from the support to be on the safe side.
4. How many chances a sequence is given depends on a countdown function that has to be carefully chosen:
   - On the one hand, holding back too much flow is a risk for producing a semimeasure with zero support.
   - On the other hand, too few attempts before giving up on sequences comes with the same risk.

Handling multiple desired properties

1. In order to handle multiple requirements, there are tasks that work independently of each other. Only once a task does not act anymore, will the next task start working.
2. **Problem:** We don't know when one task is done. So the next task might already start working, and suddenly the first task acts again. Then the second task needs to start over later.
3. So this is a finite injury construction in some sense, and this is another reason why the countdowns for each task are important.

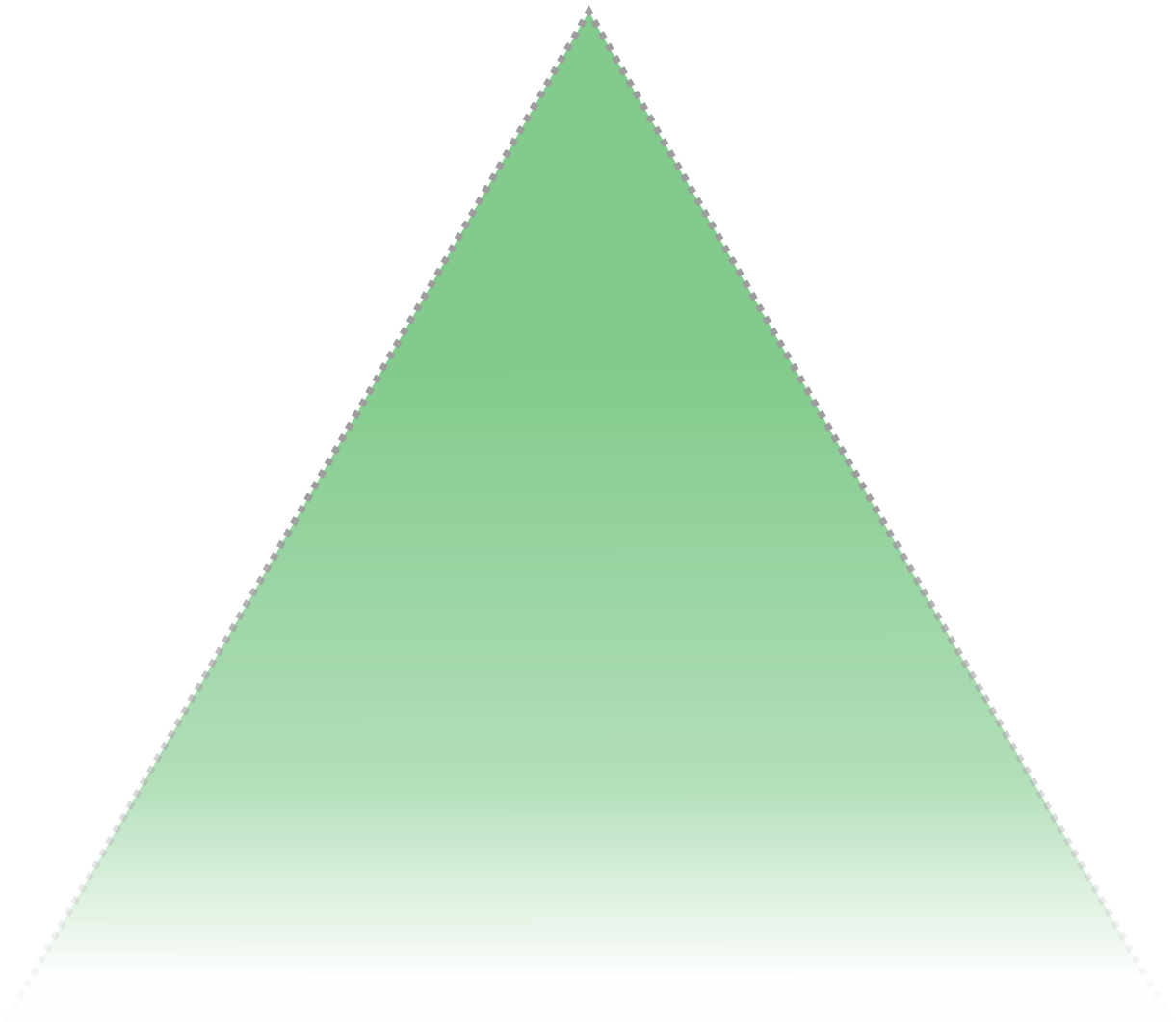
The elements of the formal construction

1. The computable *task function* $\mathfrak{t}\colon \omega \to \omega$ such that

$$\mathfrak{t}(0), \mathfrak{t}(1), \mathfrak{t}(2), \mathfrak{t}(3), \ldots = 0, 1, 0, 1, 2, 0, 1, 2, 3, 0, \ldots$$

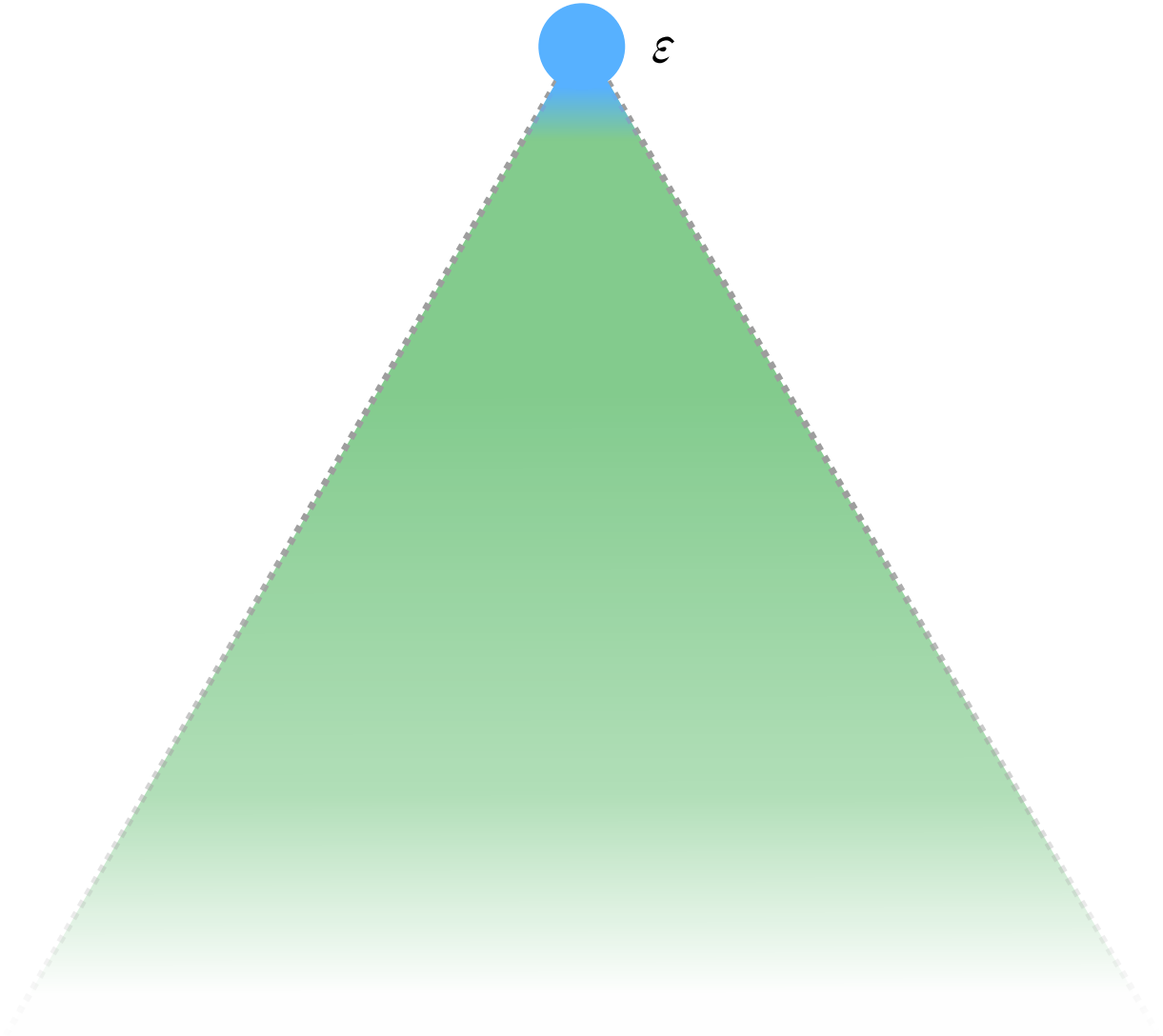
   Every node $\sigma \in 2^{<\omega}$ is assigned to task $\mathfrak{t}(|\sigma|)$.
2. A computable predicate $B(\sigma, \tau)$ that tells whether adding an extra edge from σ to τ with $\mathfrak{t}(\sigma) = \mathfrak{t}(\tau)$ helps reach our goal.

How task *i* works



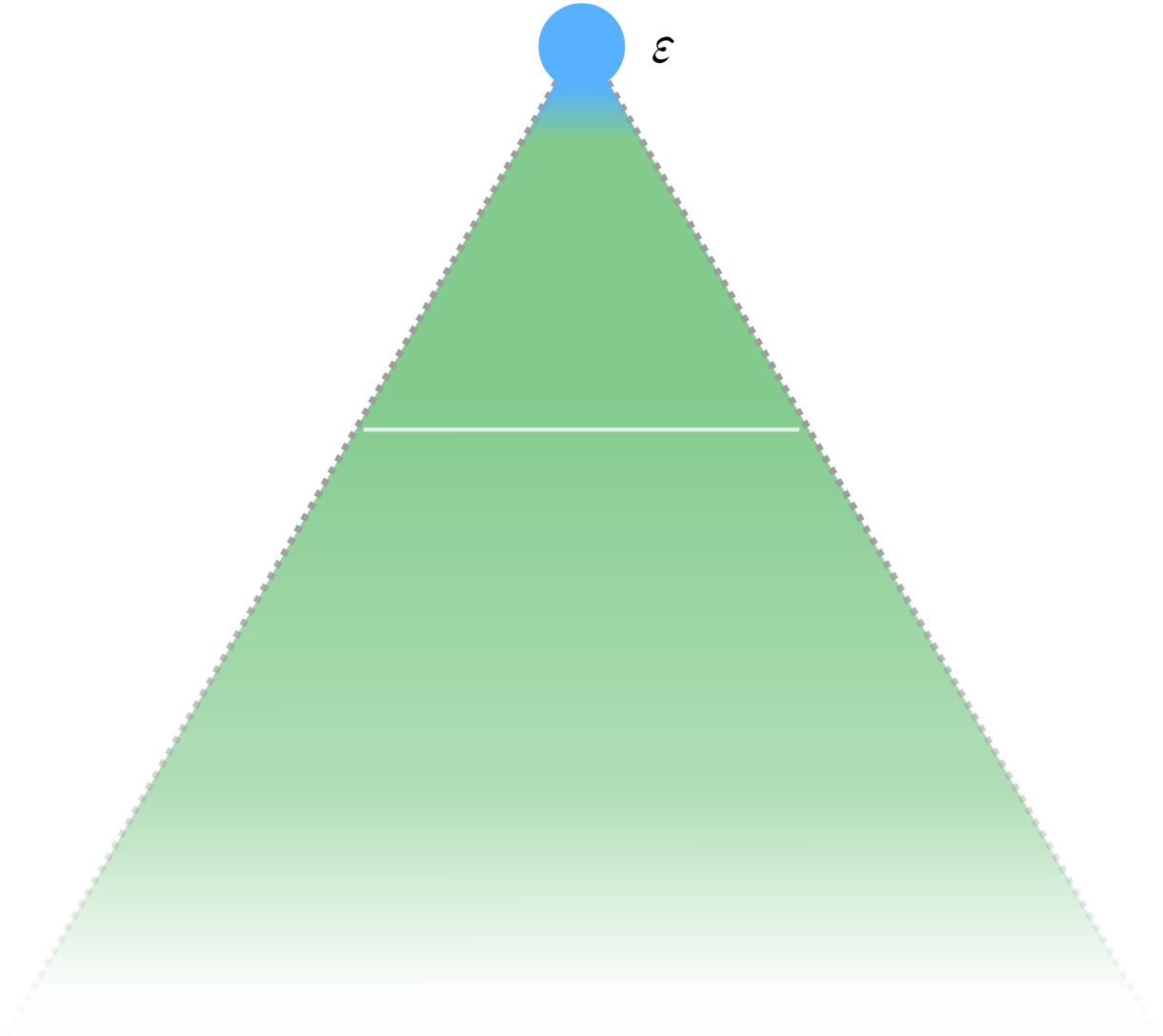
We start with Cantor space.

How task i works



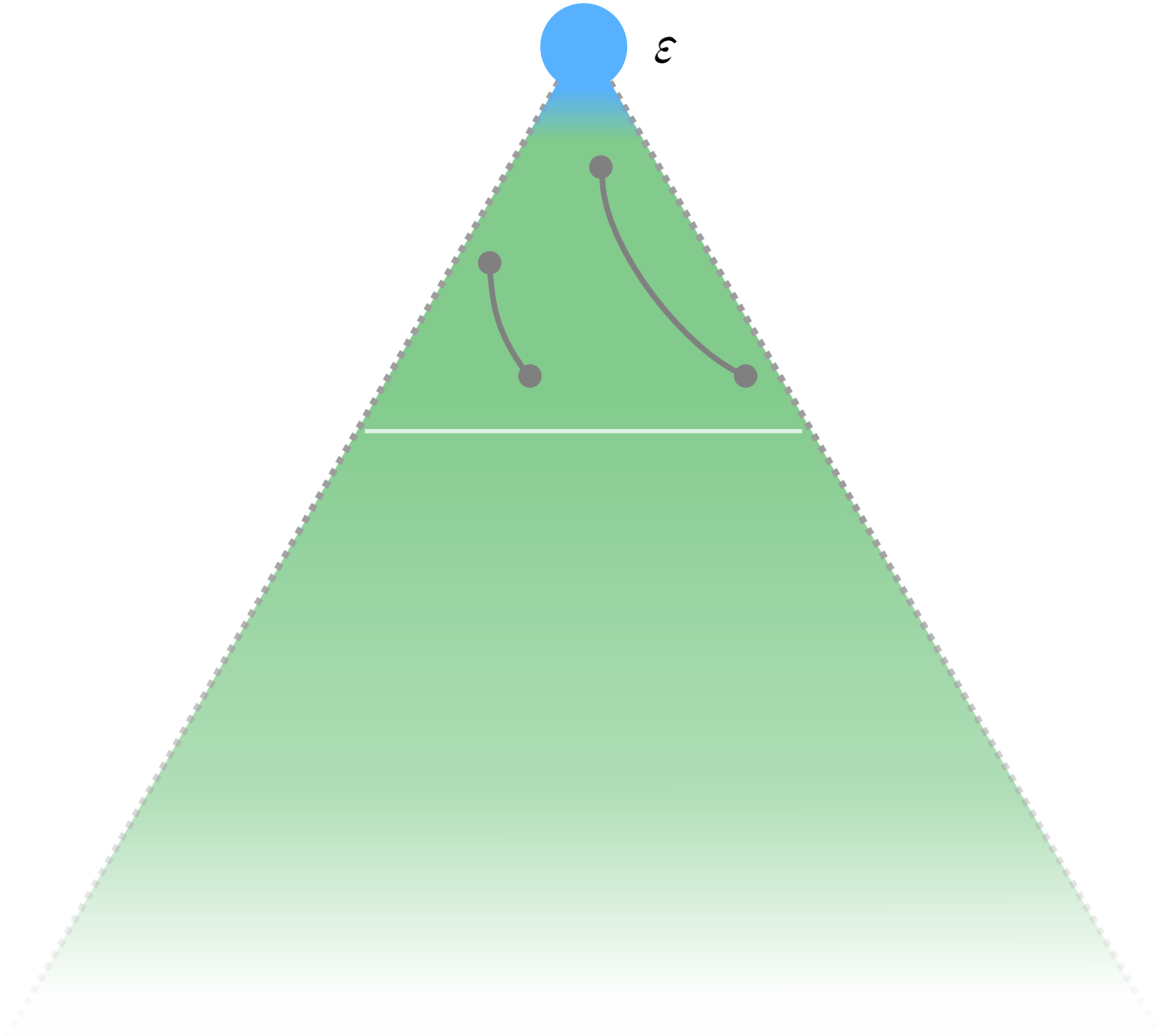
We put flow 1 at the root ε.

How task i works



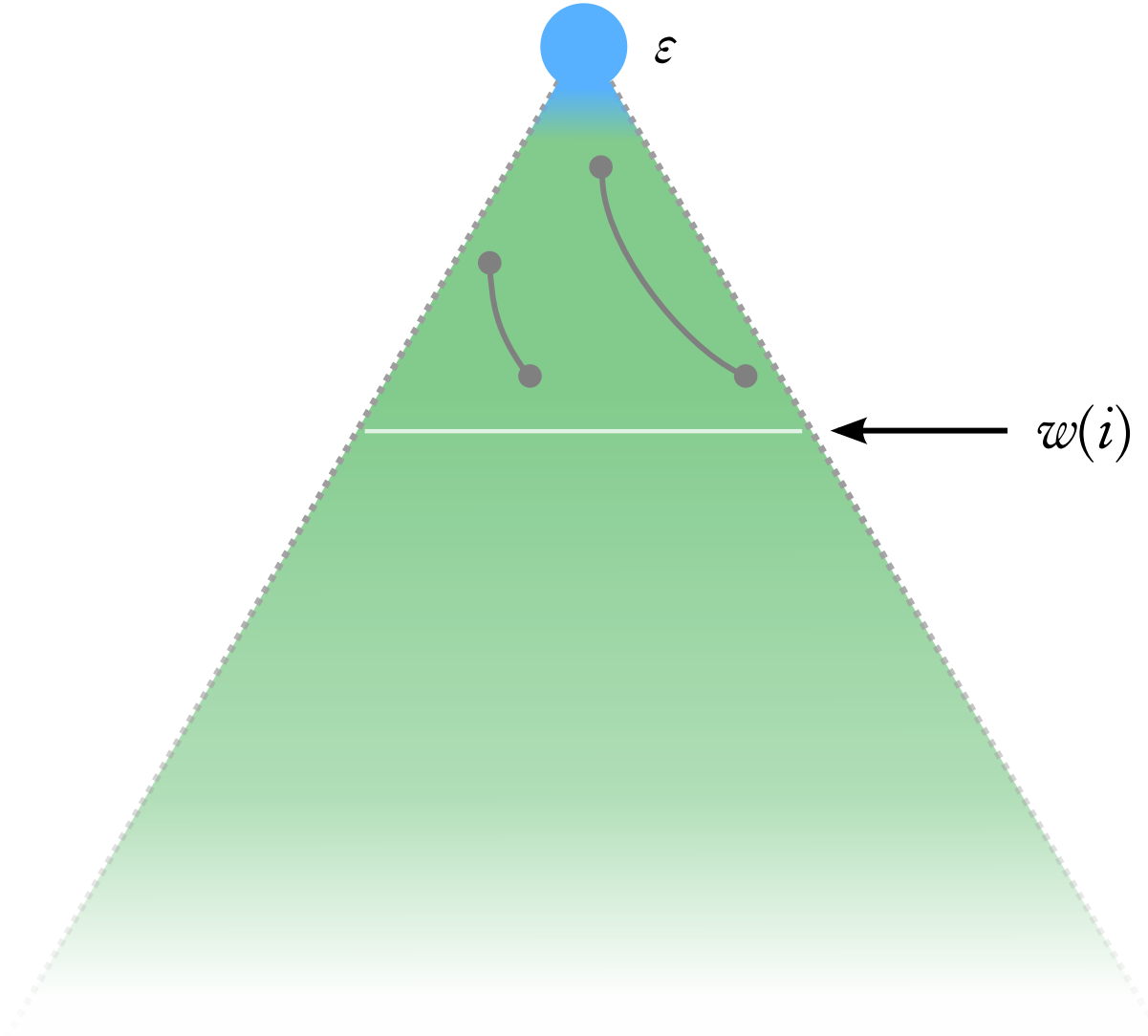
This is a node length assigned to task i.

How task i works



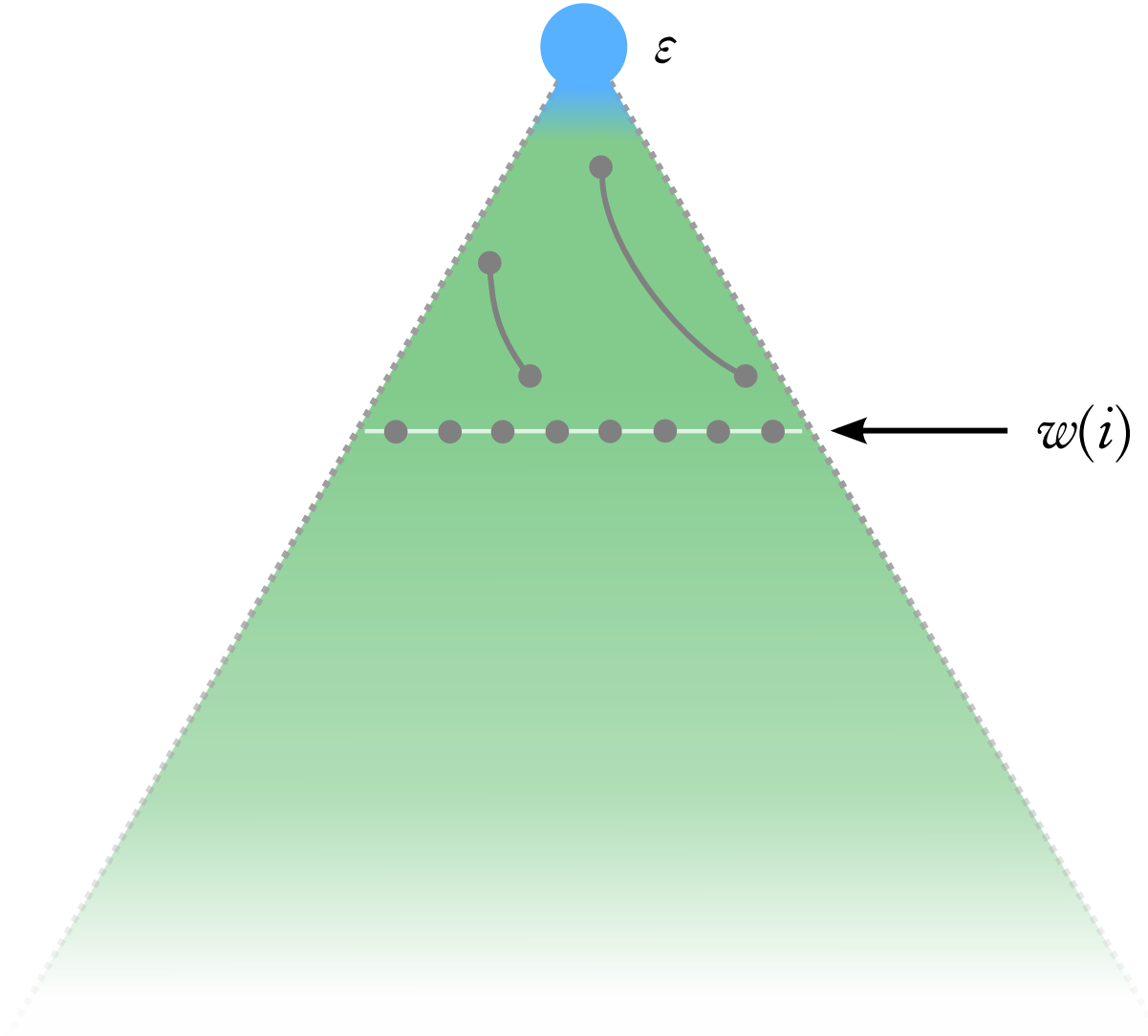
The previous task $i-1$ may have created some edges.

How task i works



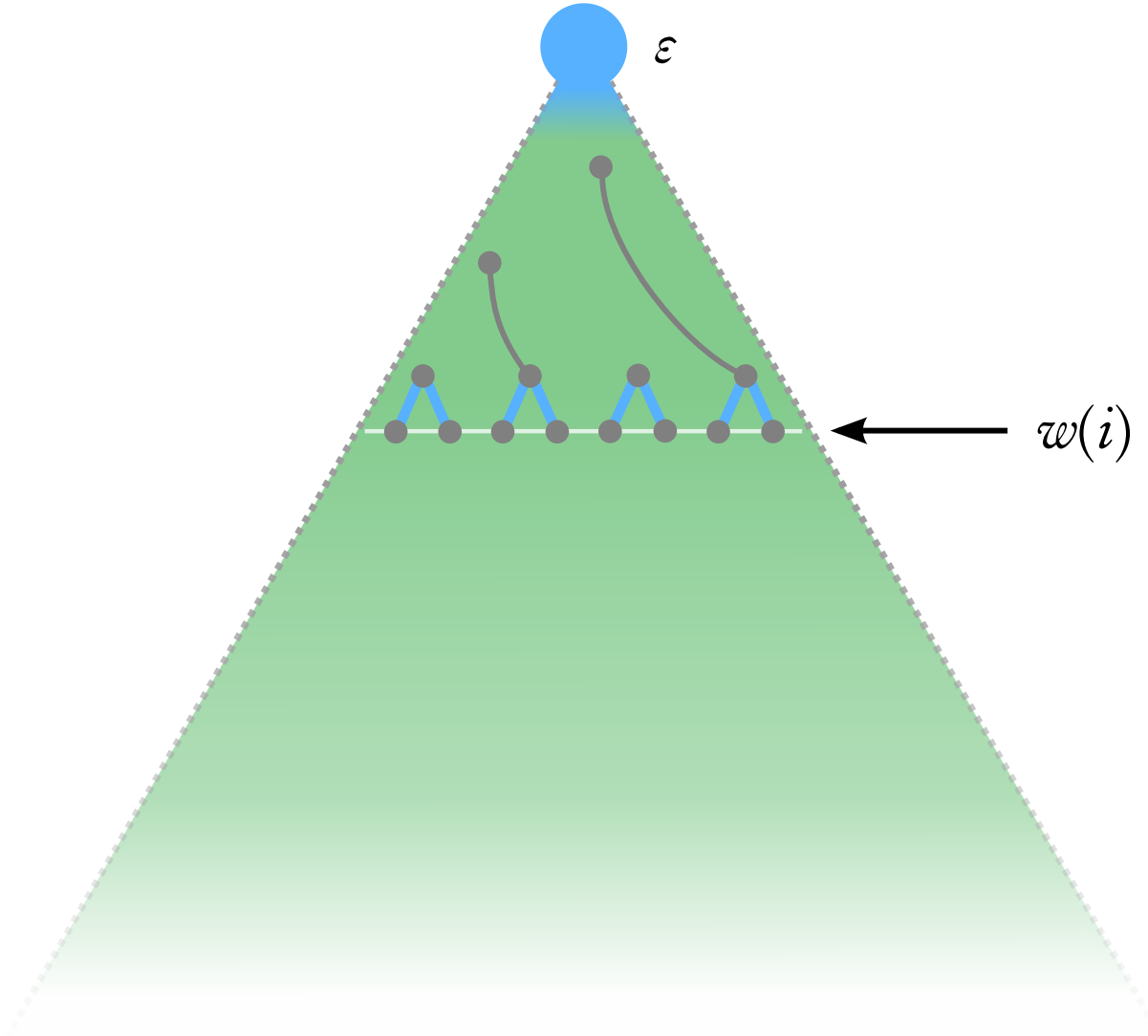
But now task $i-1$ is finished, so task i can start working on the current level to which we will refer as $w(i)$.

How task i works



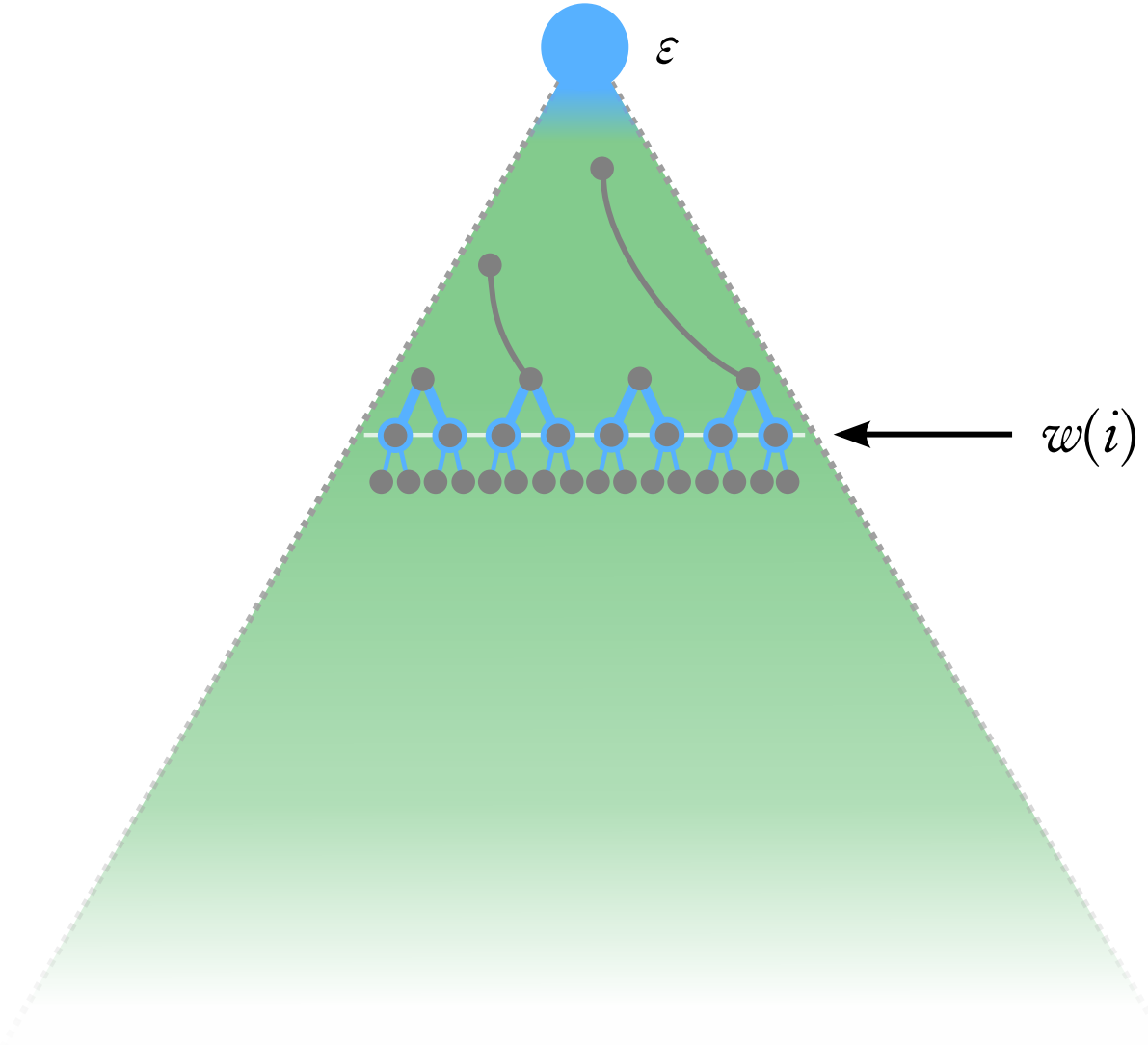
So task i can start working on the nodes of this length.

How task i works



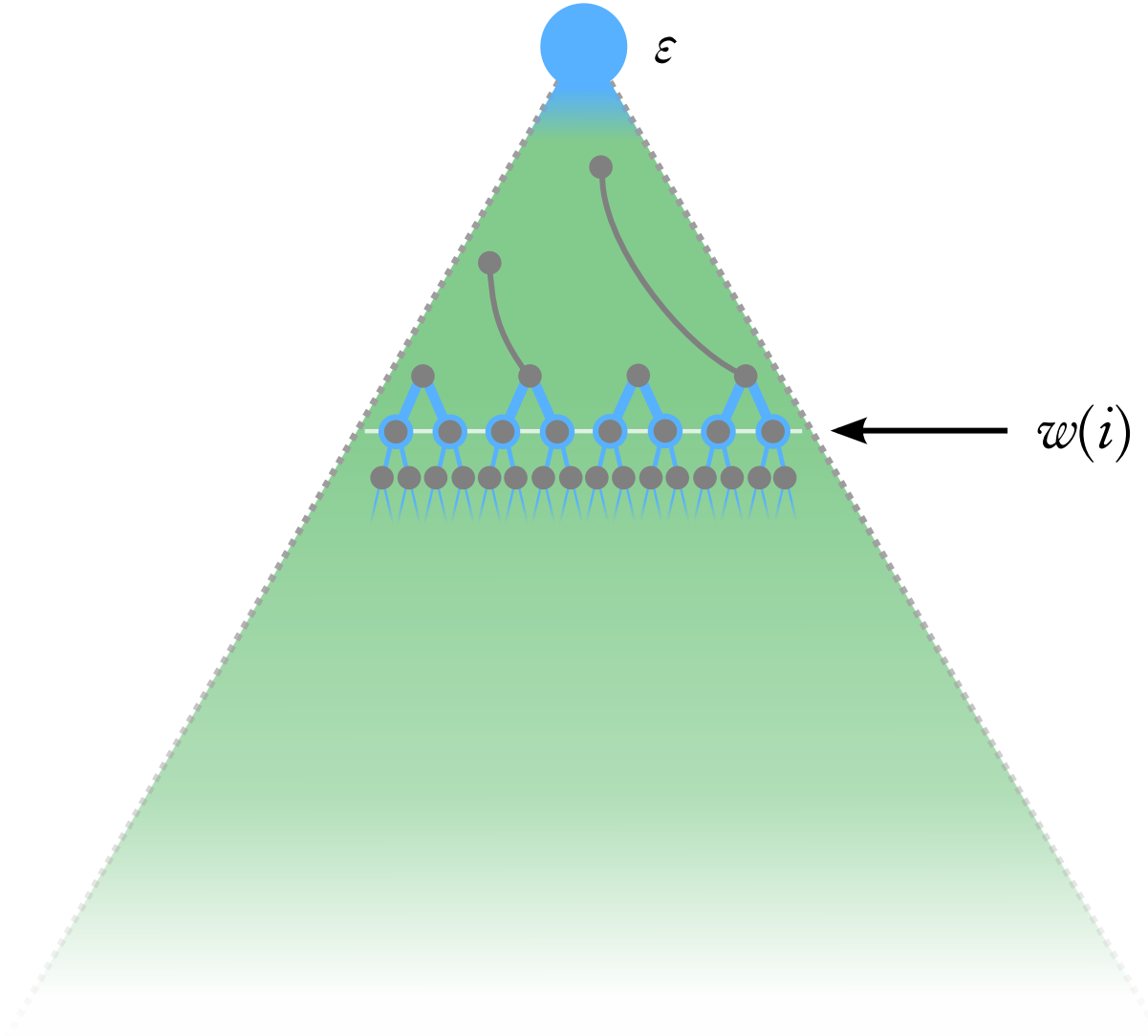
Their parents share their inflow evenly between both their children.

How task i works



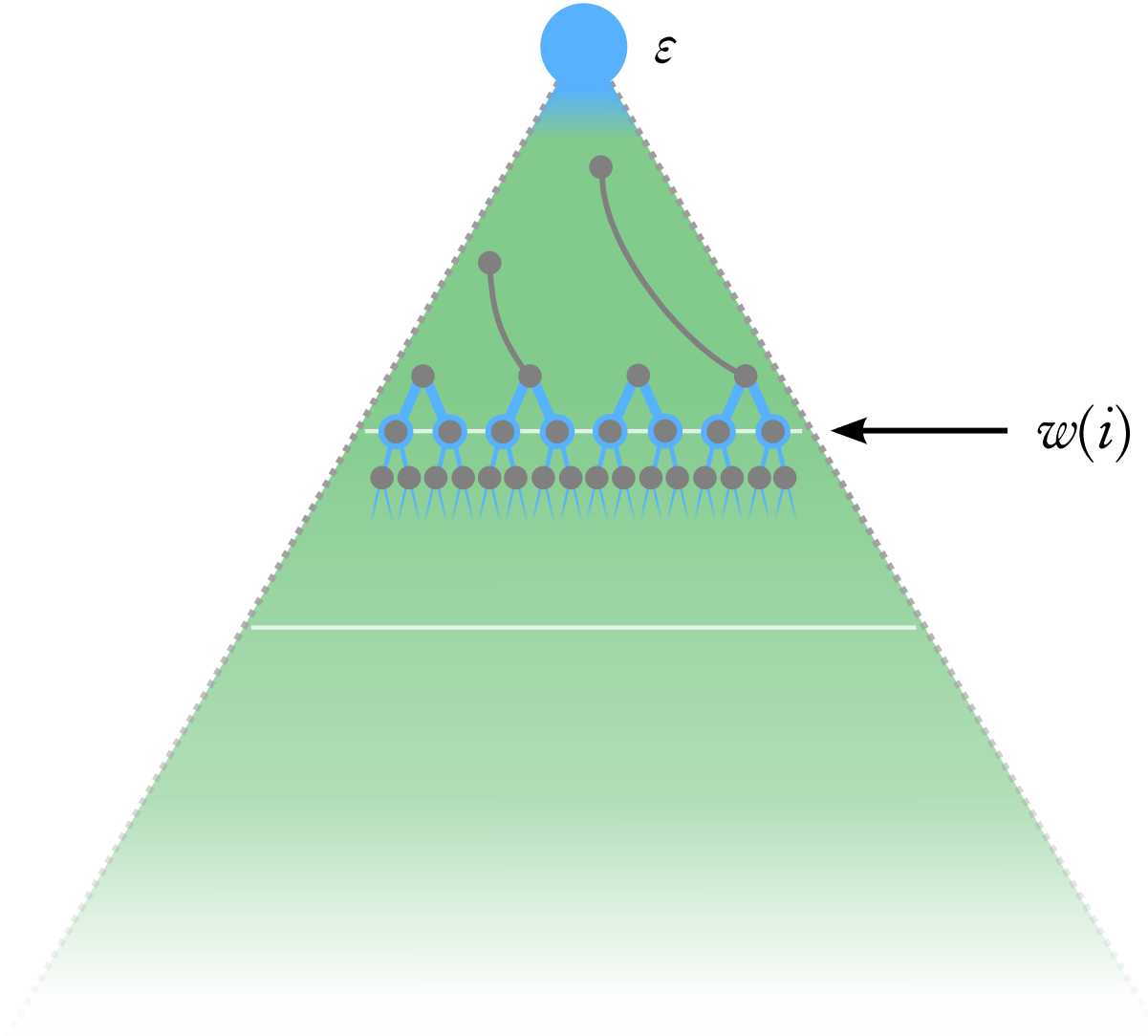
To begin the work for task i, we hold back some flow on this level, *initially activating* the nodes. The children evenly share the rest.

How task i works



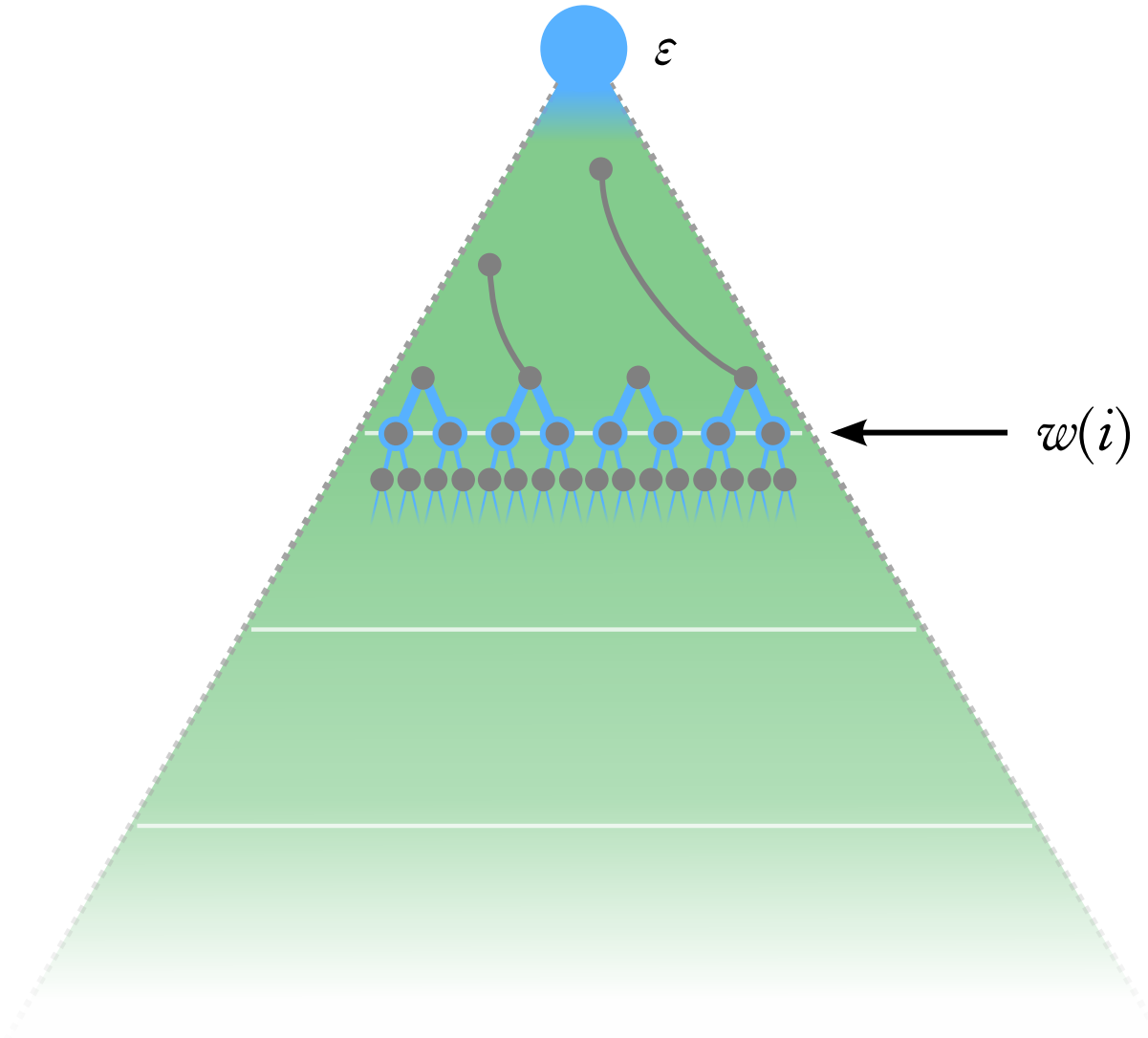
The children themselves also share their respective inflow evenly between their respective children.

How task i works



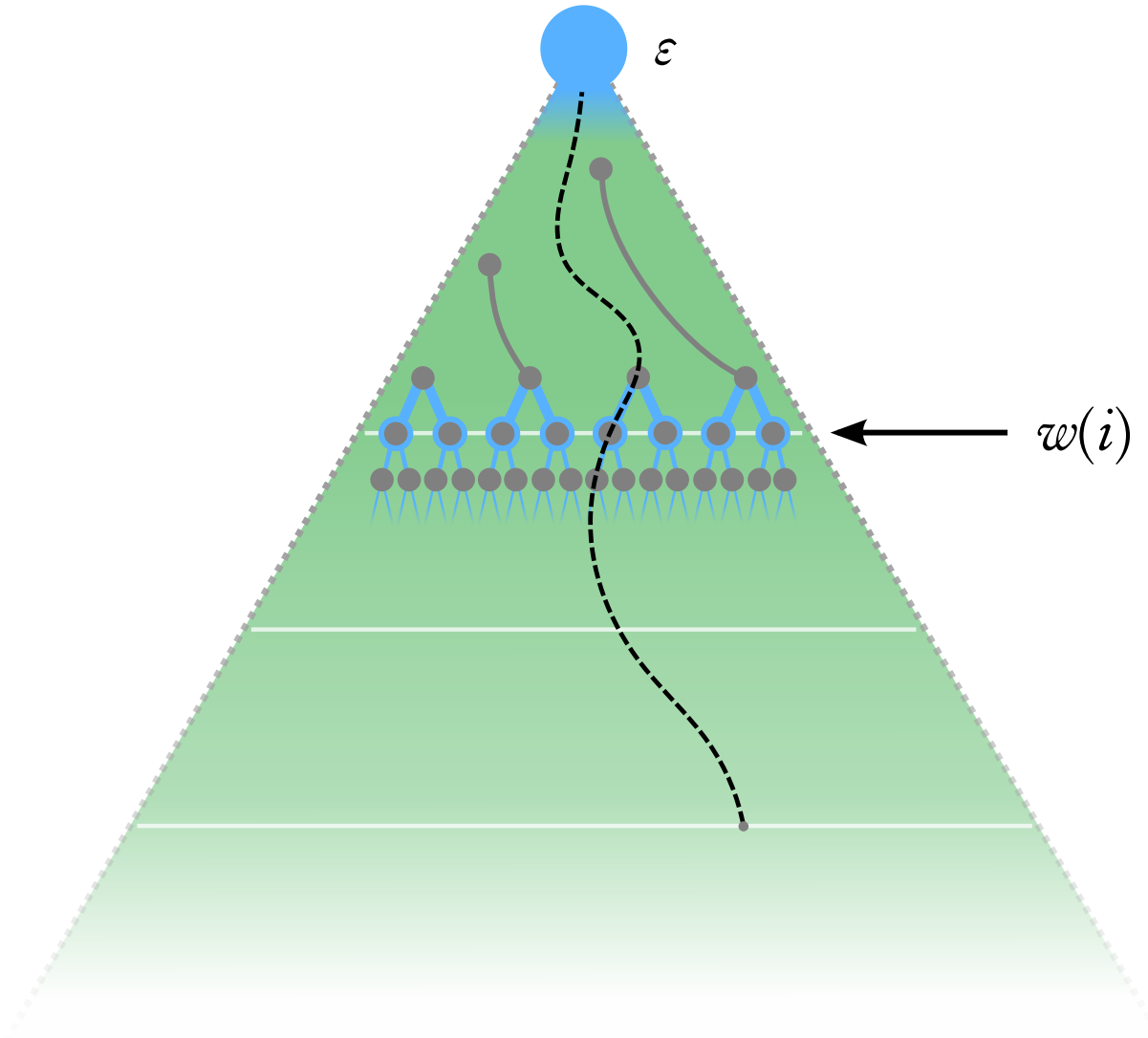
The construction continues to work. Here is another task i level, but we do not act because nowhere does predicate B hold.

How task i works



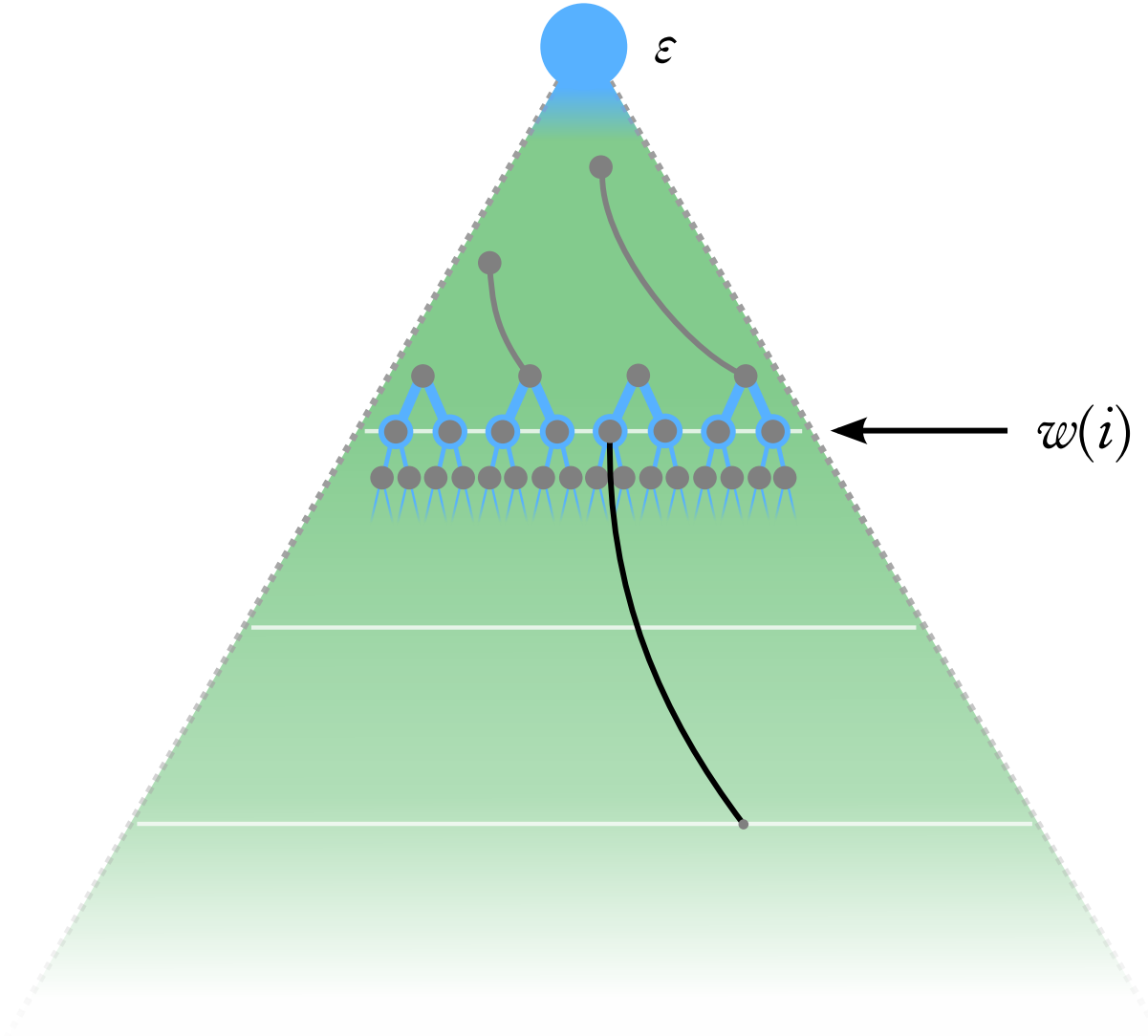
On this task i level however, we see that…

How task i works



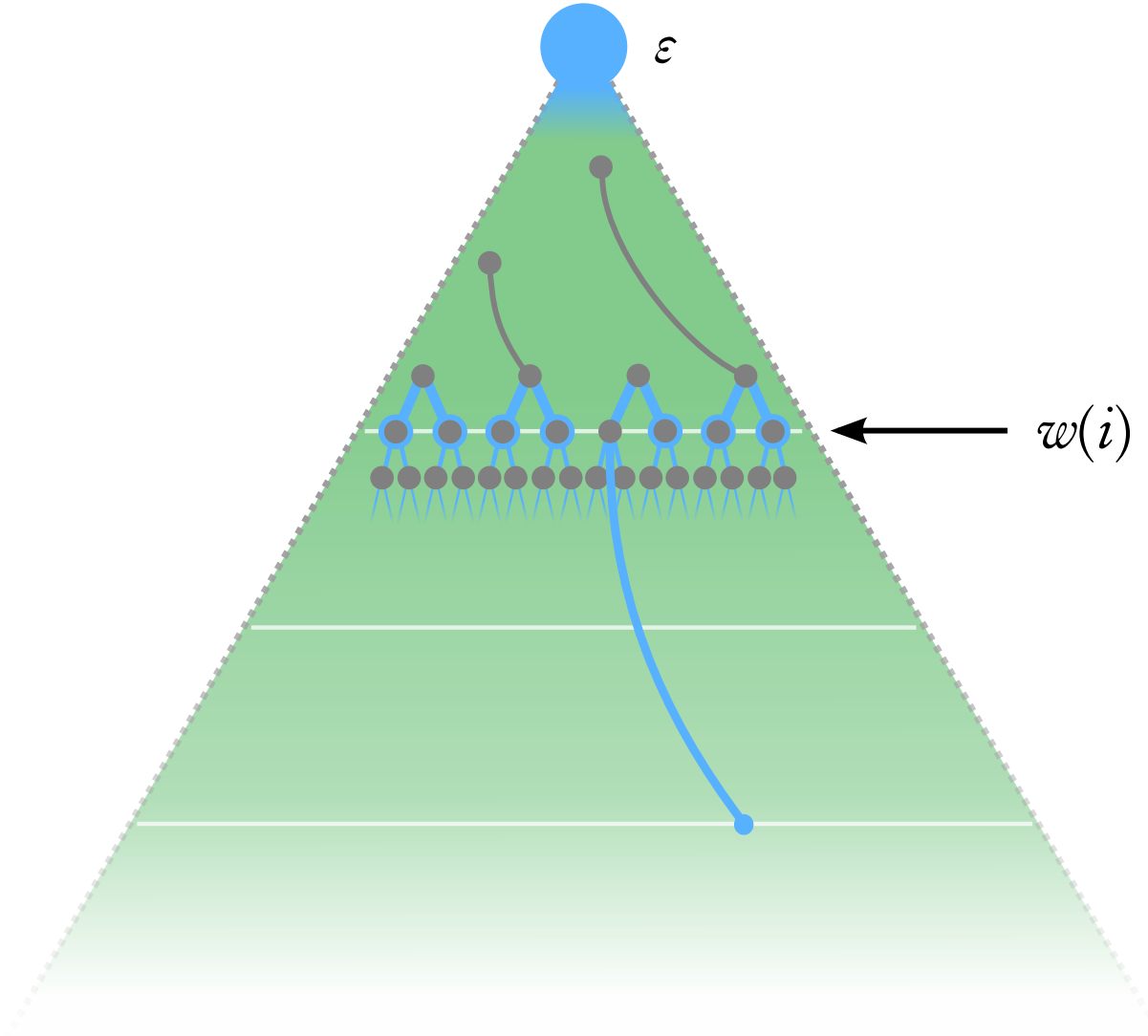
…all sequences with a certain initial segment meet the requirement associated with task i.

How task i works



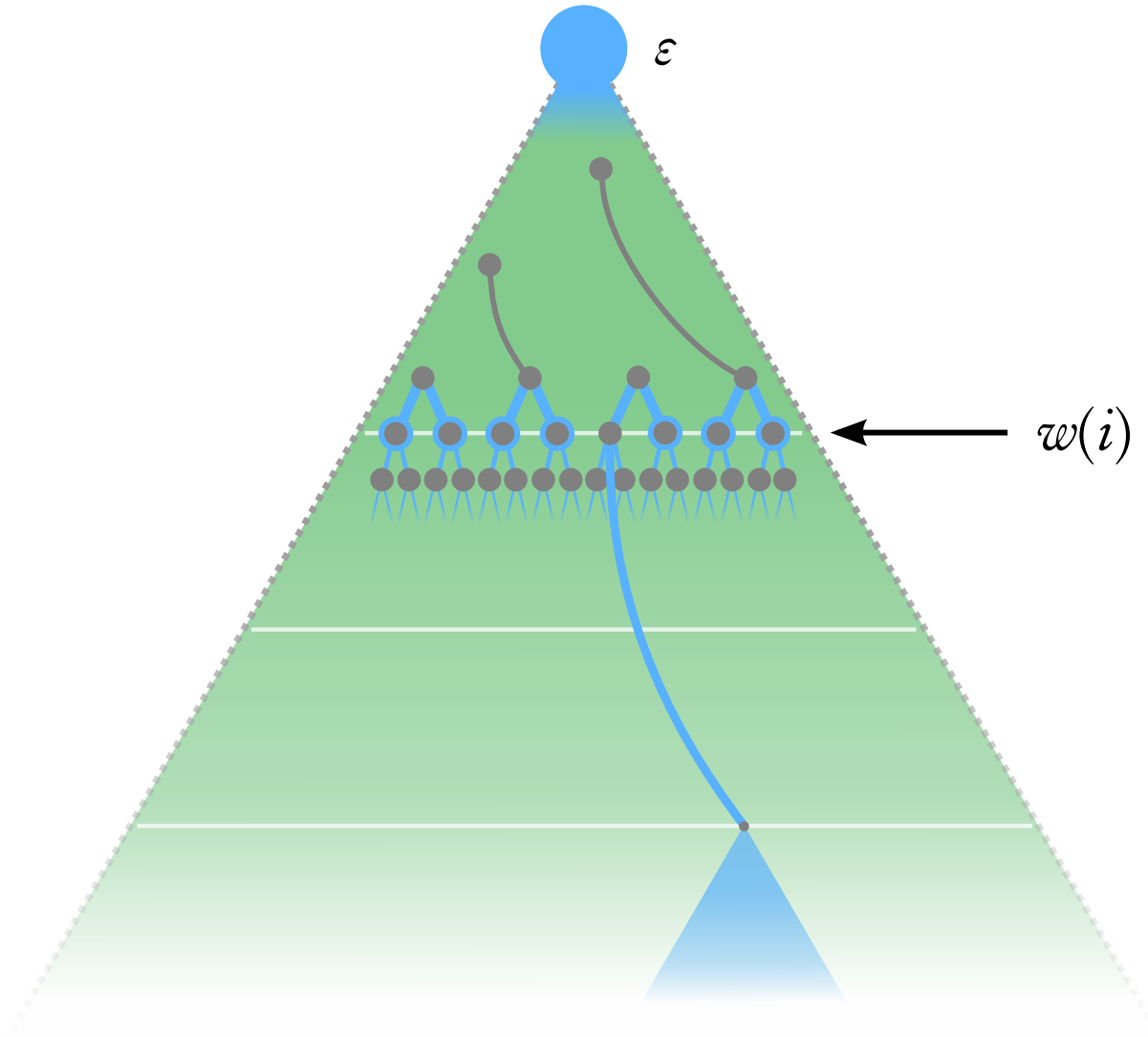
So we add a new edge to the graph…

How task i works



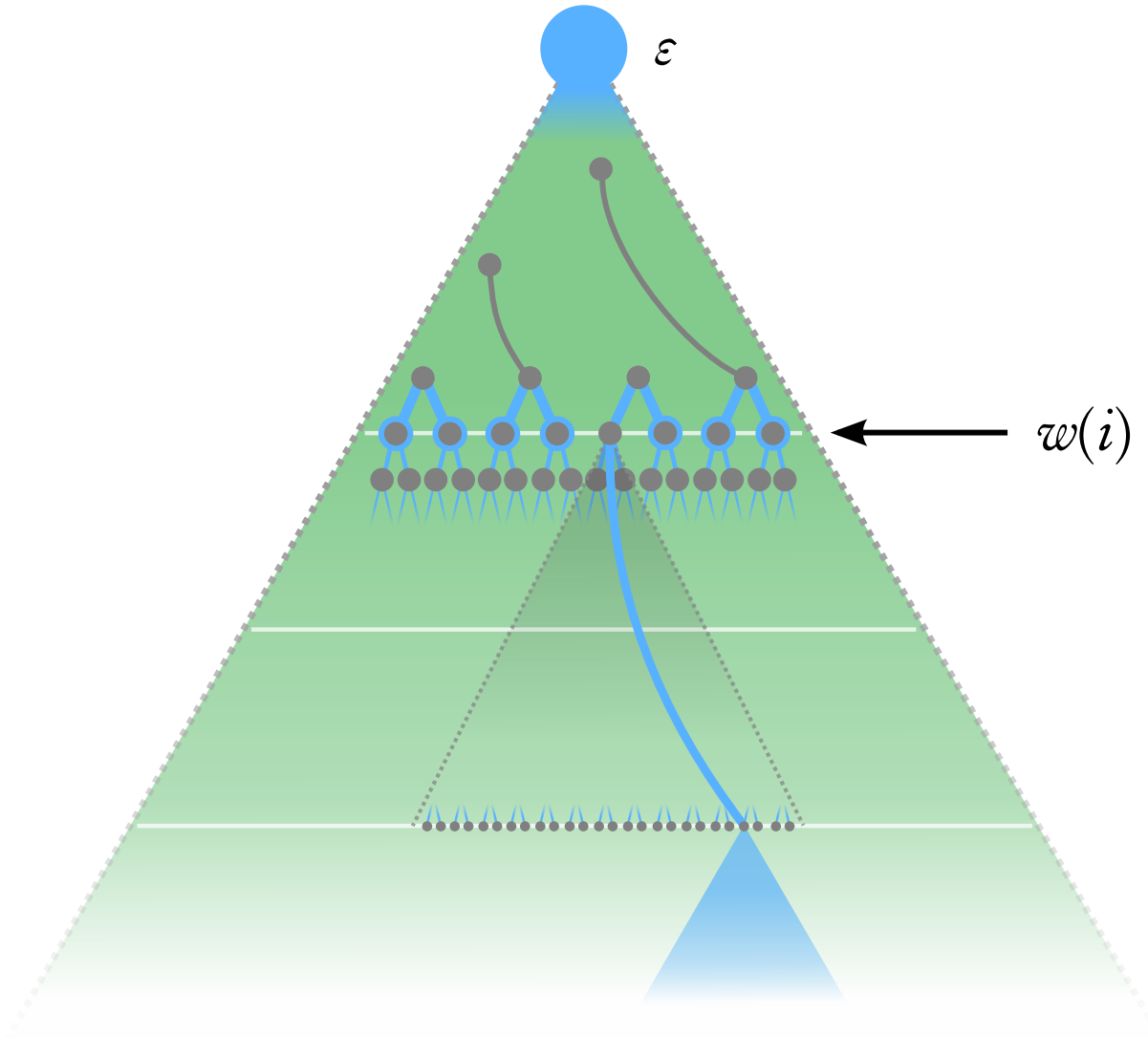
…and push the held back flow to the corresponding node.

How task i works



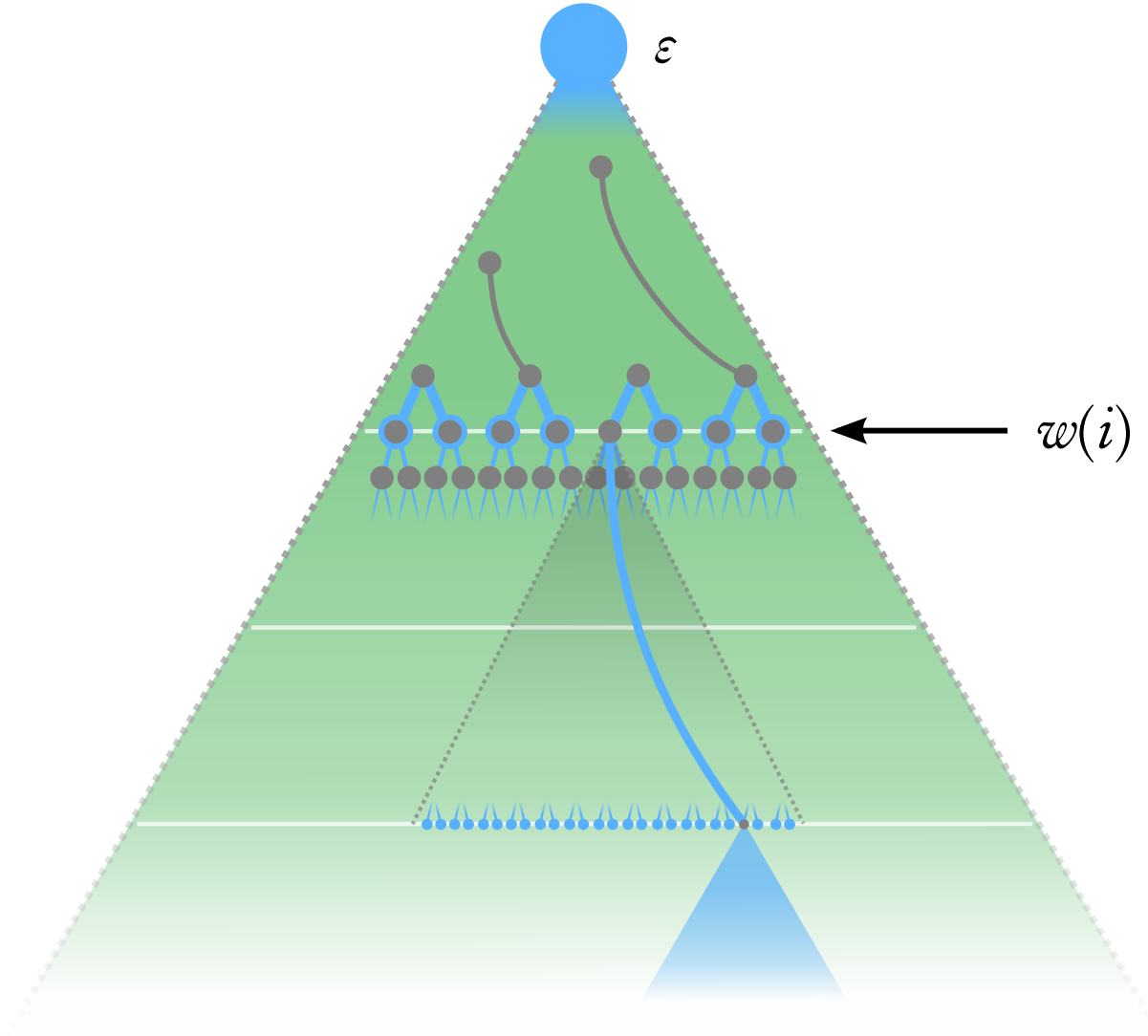
Below that node, we do not need to act anymore, so in the blue part all parent nodes share their inflow fairly between their two children.

How task i works



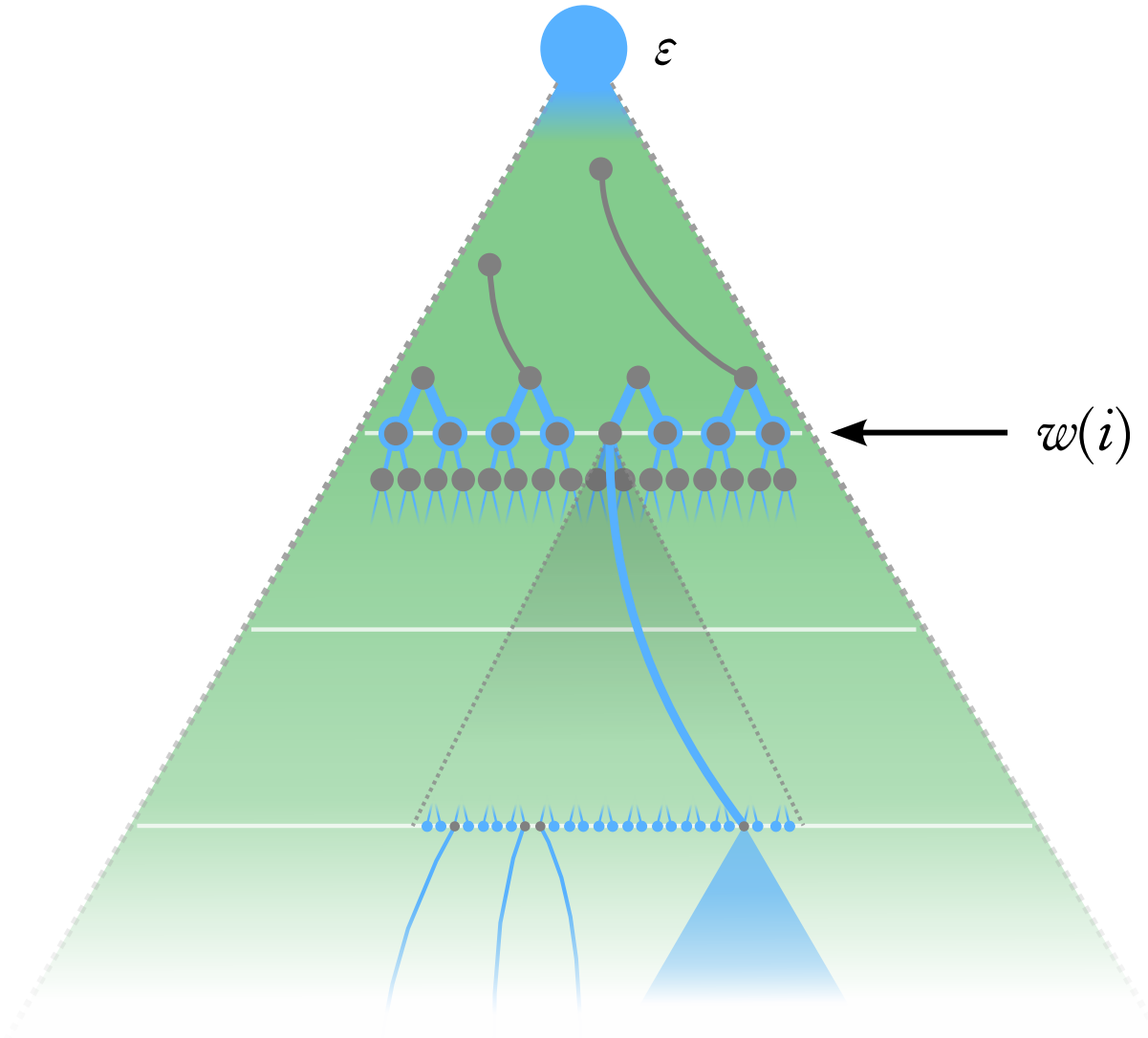
Let us look at all the nodes extending the node to which the new edge was attached. They have some inflow coming from their parents.

How task i works



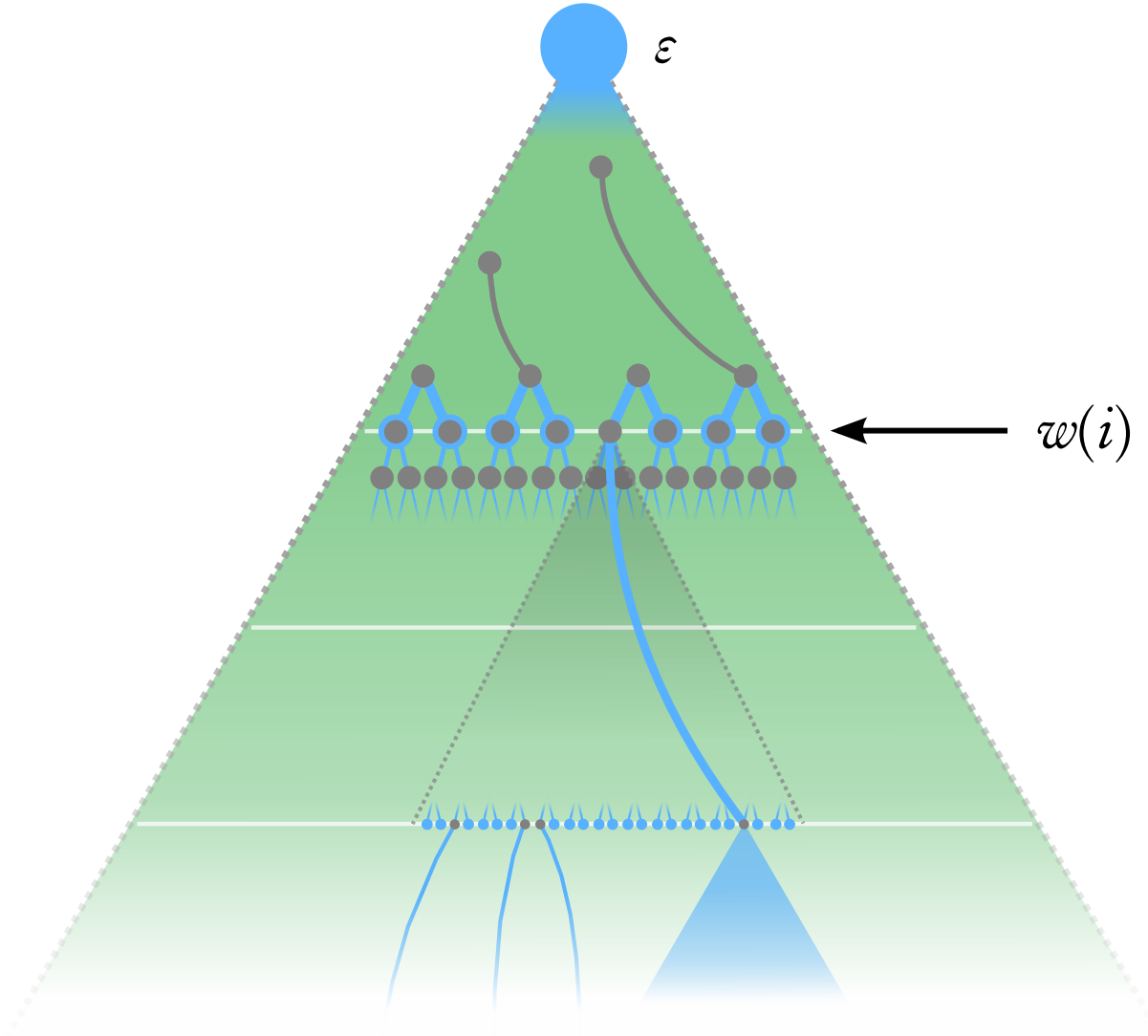
Again we decide to hold back some flow on each of them.
This is called the *noninitial activation* of these nodes.

How task i works



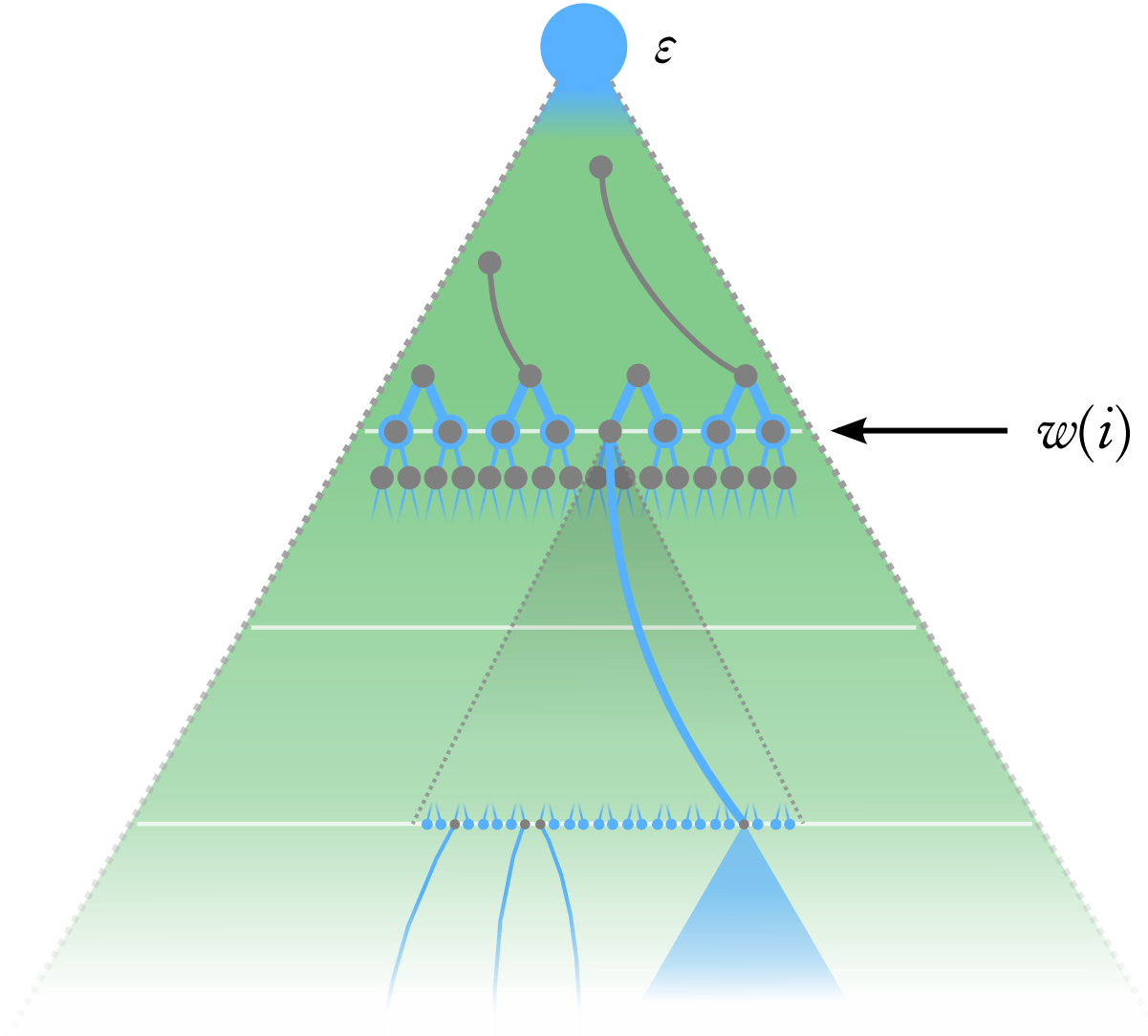
If the conditions are ever met, further edges will be attached to these later and the held back flow will be pushed through as before.

How task i works



If the conditions are never met for some node σ, the flow held back at σ is lost. Thus, holding back measure is a risky move.

How task i works



The countdown function determines, how long these chained together structures of edges can become for task i.

Properties always ensured by the construction

1. **Stability Lemma.** Every task eventually becomes inactive, so that all lower priority tasks get to act eventually.

   (This is ensured essentially due to the countdown mechanism.)

Properties always ensured by the construction

1. **Stability Lemma.** Every task eventually becomes inactive, so that all lower priority tasks get to act eventually.

   (This is ensured essentially due to the countdown mechanism.)

2. **Initial Activation Lemma.** If we choose the amounts risked at initially activated nodes cleverly, then we can control how much flow can in the worst case be lost because of measure held back and then never used.

   (This follows from a calculation.)

Properties always ensured by the construction

1. **Stability Lemma.** Every task eventually becomes inactive, so that all lower priority tasks get to act eventually.

   (This is ensured essentially due to the countdown mechanism.)

2. **Initial Activation Lemma.** If we choose the amounts risked at initially activated nodes cleverly, then we can control how much flow can in the worst case be lost because of measure held back and then never used.

   (This follows from a calculation.)

3. **Noninitial Activation Lemma.** On nodes that get noninitially activated, we don't lose too much either.

   (It is enough to look at the total amount of flow that arrives in all nodes of equal length through normal edges as the last step. It suffices to show that this quantity never decreases due to noninitial risking. The argument for this is that noninitial activation occurs only as a sideeffect of an edge having been added. Then a calculation shows that the flow that was "secured" by that edge is larger than the total flow risked as a result.)

Properties always ensured by the construction

1. **Measure Lemma.** The construction ensures that the support does not become zero.

   (Follows essentially from the Initial and Noninitial Activation Lemmata.)

Properties always ensured by the construction

1. **Measure Lemma.** The construction ensures that the support does not become zero.

   (Follows essentially from the Initial and Noninitial Activation Lemmata.)

2. **Continuity Lemma.** The semimeasure has no atoms.

   (This is because, for σ where $|\sigma| = w(i) - 1$ for some $i < \omega$, there cannot be any extra edges starting at σ and all flow out of σ is evenly split between $\sigma 0$ and $\sigma 1$.)

Properties always ensured by the construction

1. **Measure Lemma.** The construction ensures that the support does not become zero.

   (Follows essentially from the Initial and Noninitial Activation Lemmata.)

2. **Continuity Lemma.** The semimeasure has no atoms.

   (This is because, for σ where $|\sigma| = w(i) - 1$ for some $i < \omega$, there cannot be any extra edges starting at σ and all flow out of σ is evenly split between $\sigma 0$ and $\sigma 1$.)

3. **Effective closure.** The support is Π^0_1.

   (As sequences only get removed due to decisions made at some construction step.)

The verification necessary for the specific use

1. A sequence which has on itself an edge for each task is good, since that means it meets all requirements.
2. **Dichotomy.** Thus, we need to ensure that on every sequence
   - either the sequence is "put in the bank" for *each* task by putting an edge for each task on the sequence,
   - or the countdown runs out for *at least one* task, and the sequence is removed from the support.
3. Then we can be sure that each sequence in the support has the desired computability-theoretic properties.

4 Applications

A first application

1. **Theorem (V'yugin, restated).** There is a left-c.e. semimeasure P such that
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X) \notin \mathrm{MLR}$.
2. *Note how the first three items are now immediate from the construction technique and require no further proof!*
3. We only need to argue the application-specific fourth item.

A first application

1. **So we need to show:** For each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X) \notin \mathrm{MLR}$.
2. Let $\langle j, s \rangle$ be the task associated with nodes $\sigma \prec \tau$. Then

$$B(\sigma, \tau) \text{ holds } \quad :\Longleftrightarrow \quad \Phi^{\tau}_{j,|\tau|} \text{ is very long,}$$

   where larger s or longer σ means longer.
3. Then we can define a Martin-Löf test by letting, for $s \in \omega$,

$$\mathscr{U}_s = \bigcup_{\substack{n \in \omega,\, \mathrm{t}(n) = \langle j,s \rangle, \\ \beta \in 2^{<\omega}[\langle j,s \rangle],\, |\beta| = n}} [\![\Phi^{\beta}_{j,n}]\!].$$

   By the choice of B, the measure of this test is small enough.

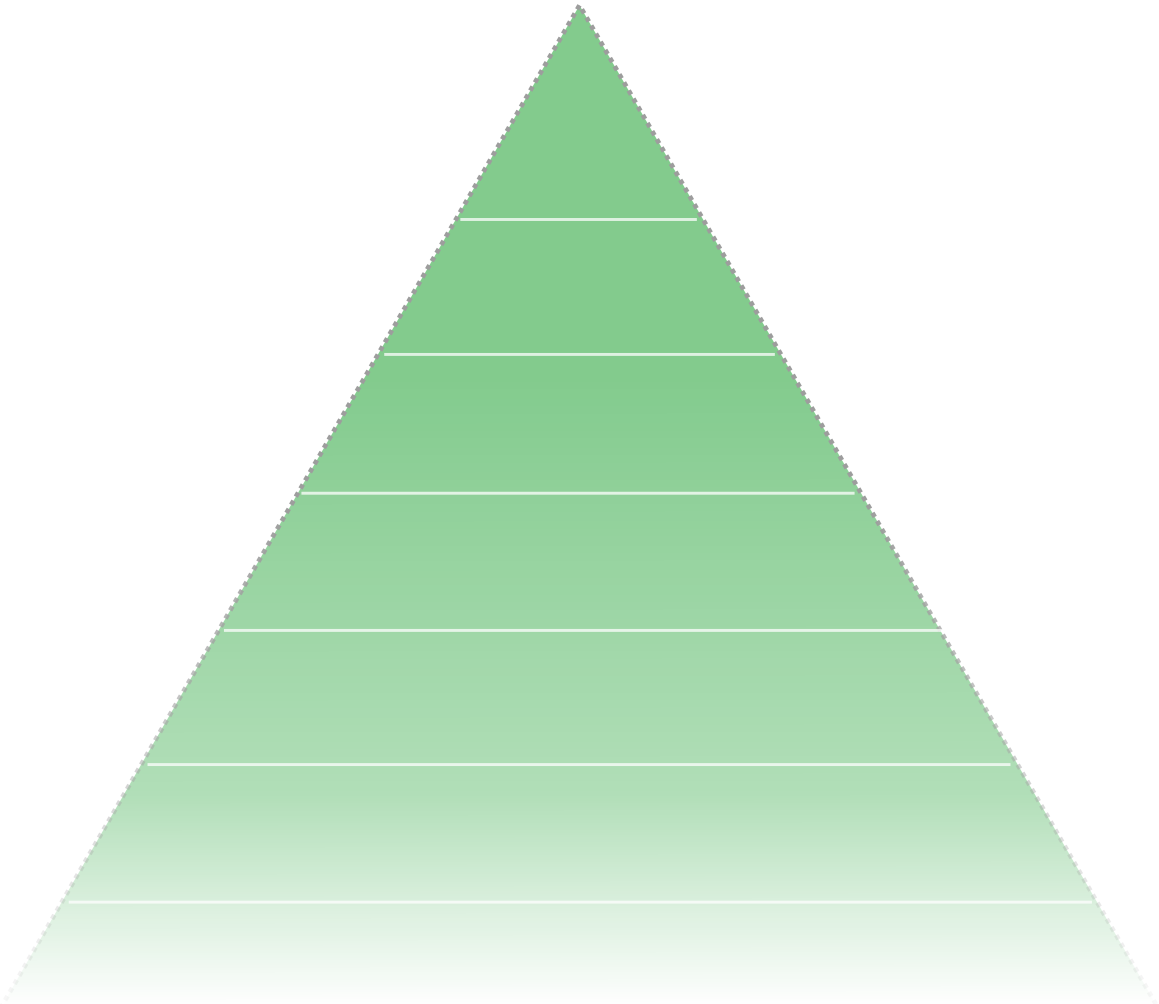
The dichotomy holds

1. It remains to argue that the dichotomy holds for this choice of B.
2. So if a sequence does not meet a requirement, we need to be sure that it will definitely be removed from the support.

The dichotomy holds

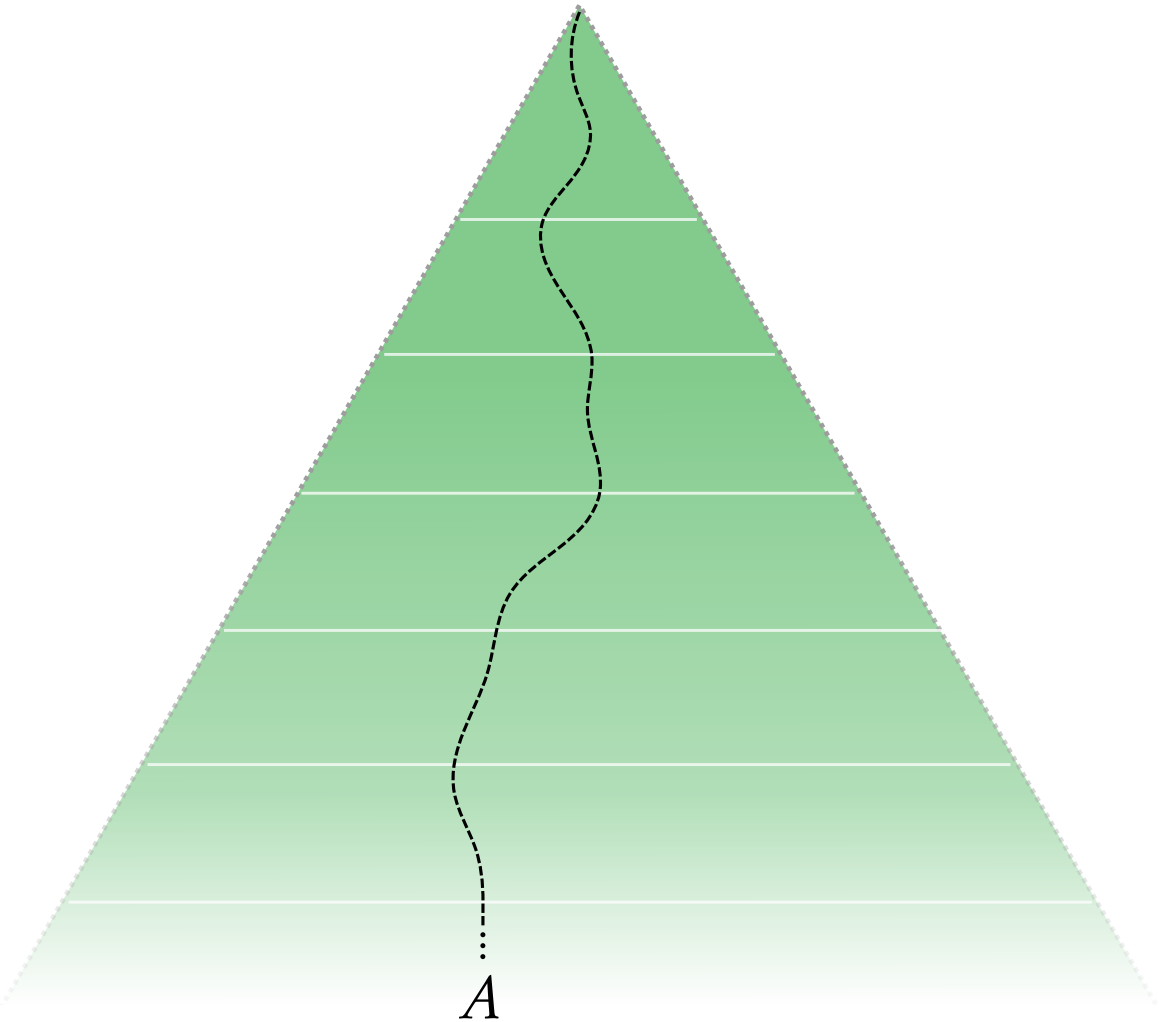
1. It remains to argue that the dichotomy holds for this choice of B.
2. So if a sequence does not meet a requirement, we need to be sure that it will definitely be removed from the support.
3. How does the removal work and what could go wrong?

The dichotomy holds



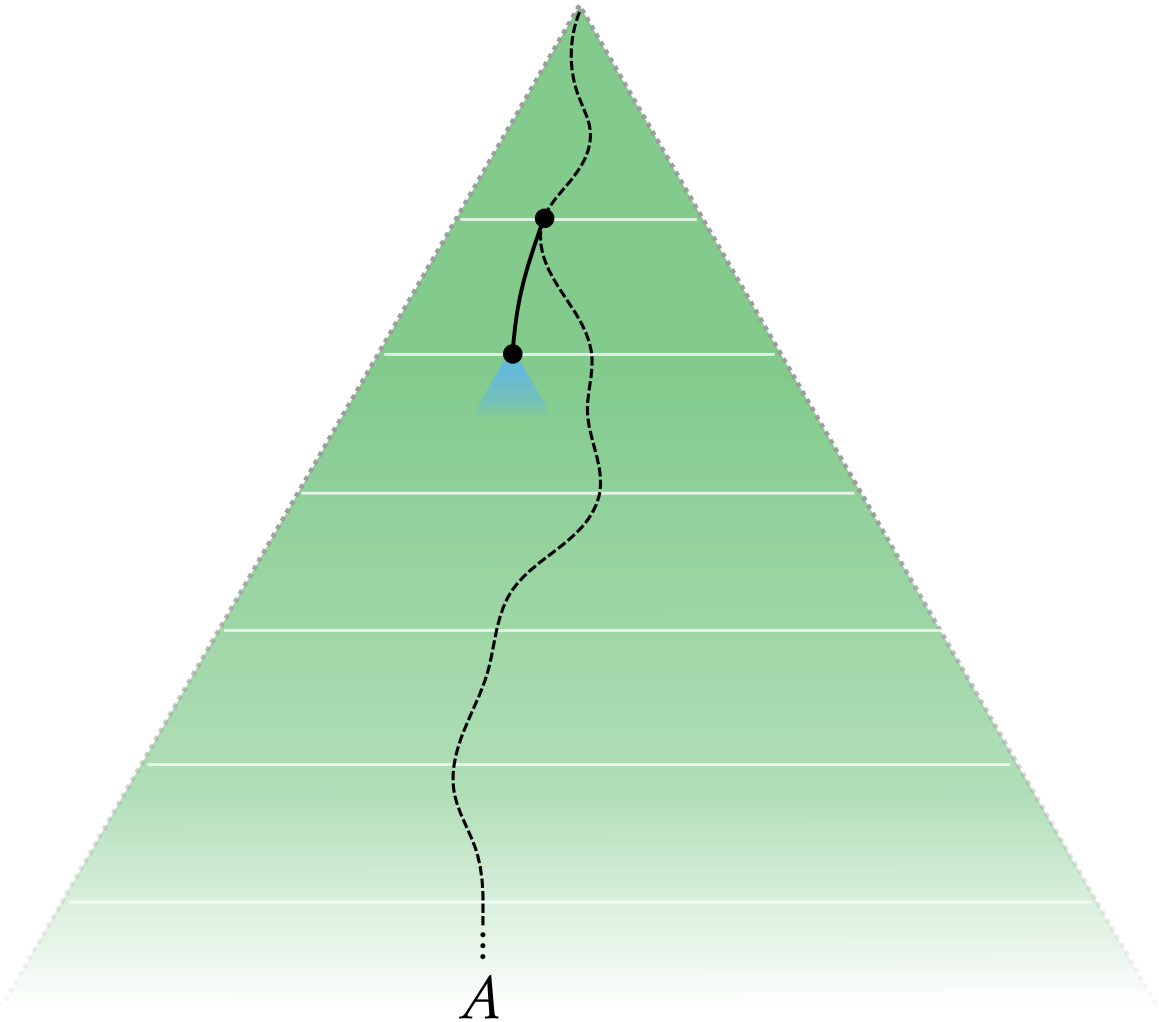
Again we look at Cantor space.

The dichotomy holds



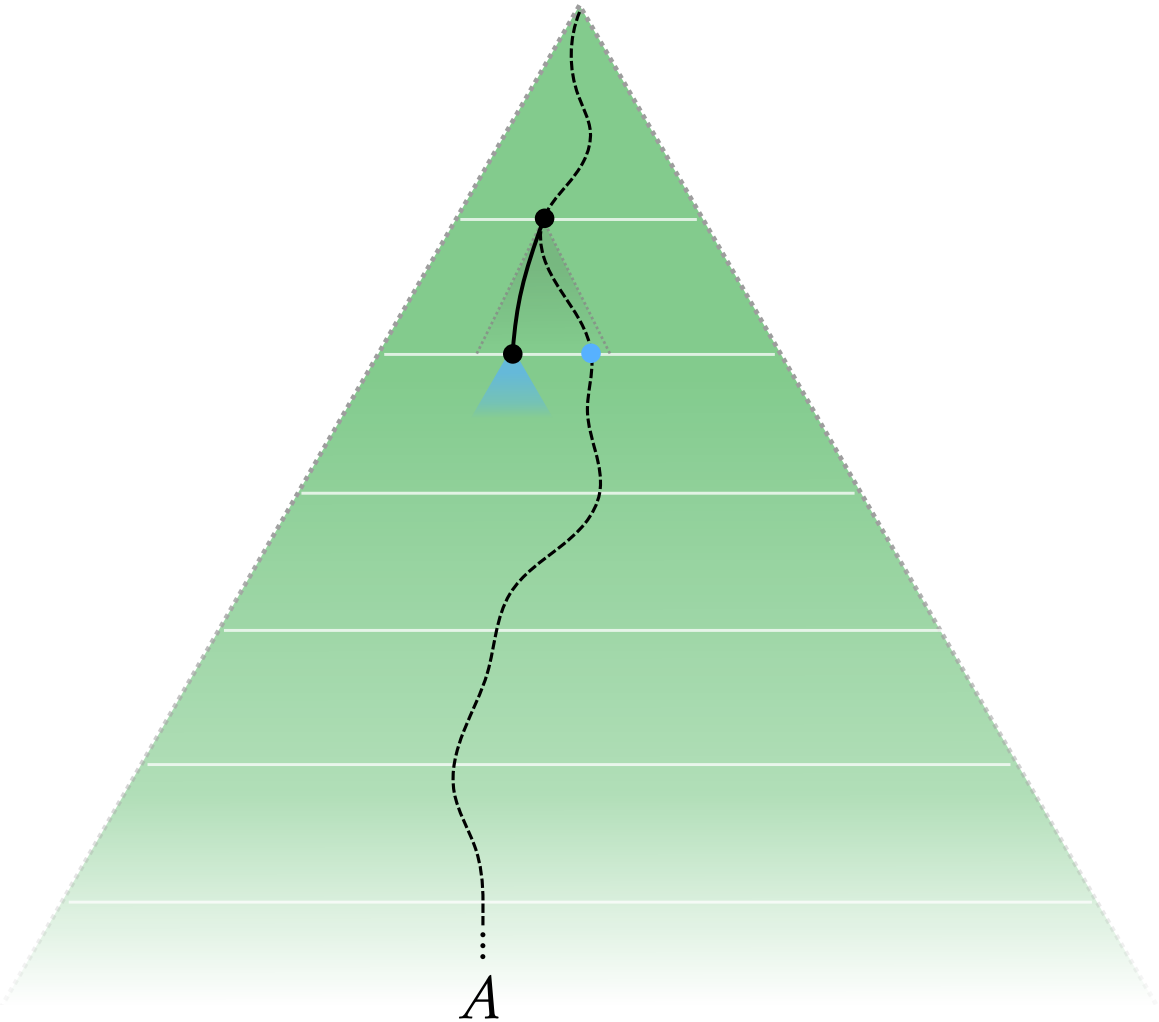
Assume that A does not meet the requirement assigned to task i.
What needs to happen for it to get removed from the support?

The dichotomy holds



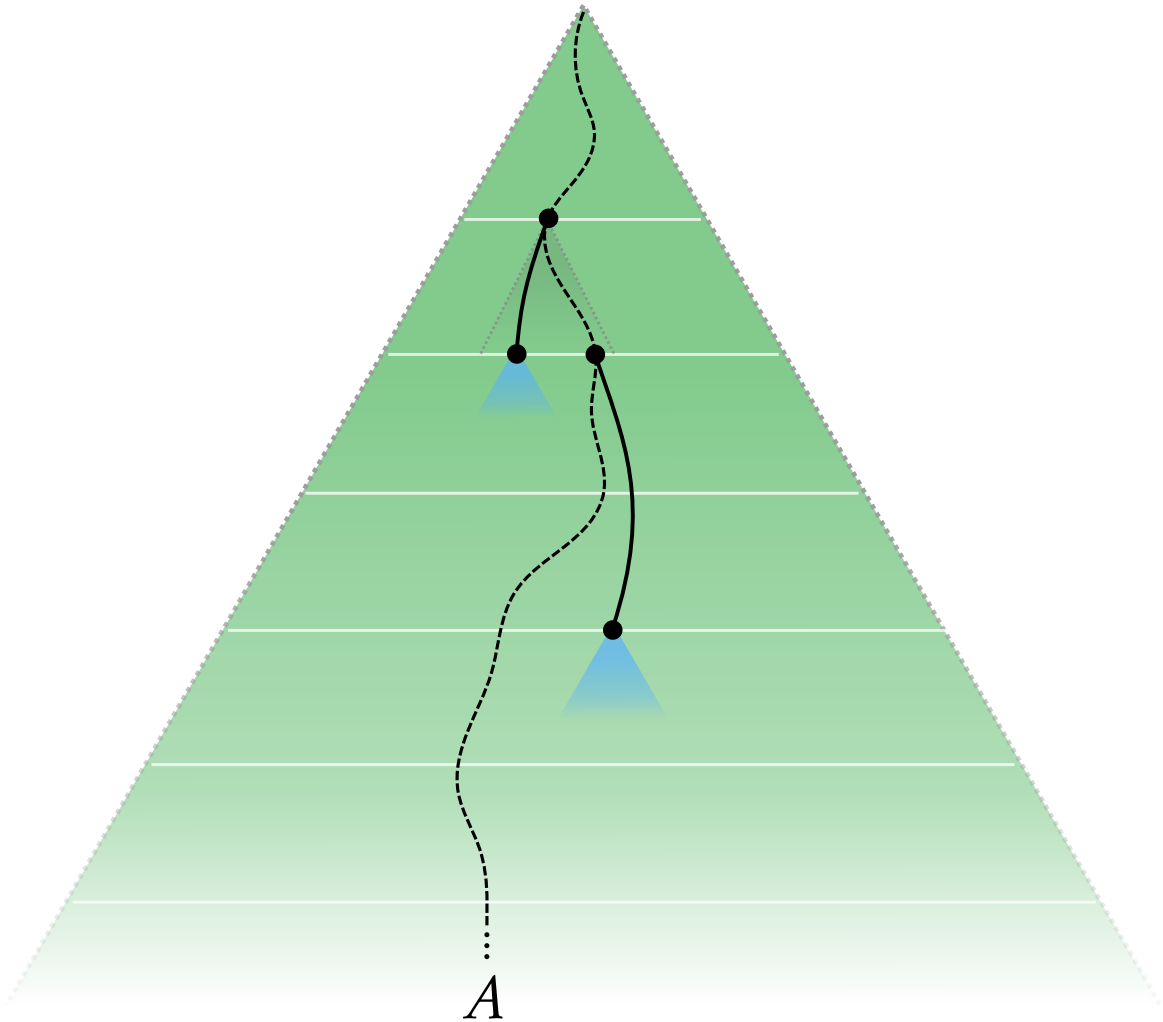
Some new edge branches off A.

The dichotomy holds



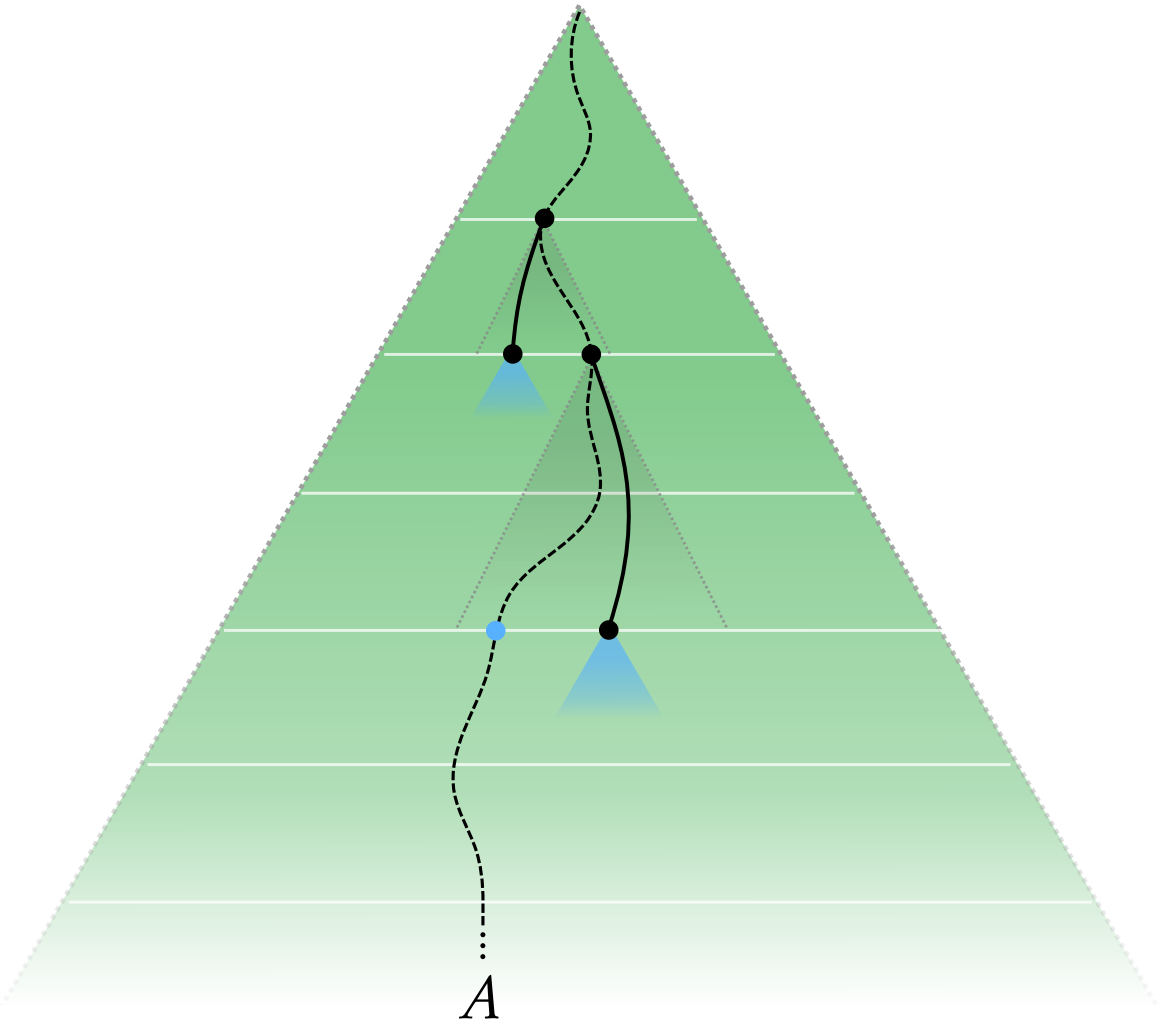
As discussed, that means that a longer initial segment of A stays active, but with a lower countdown value.

The dichotomy holds



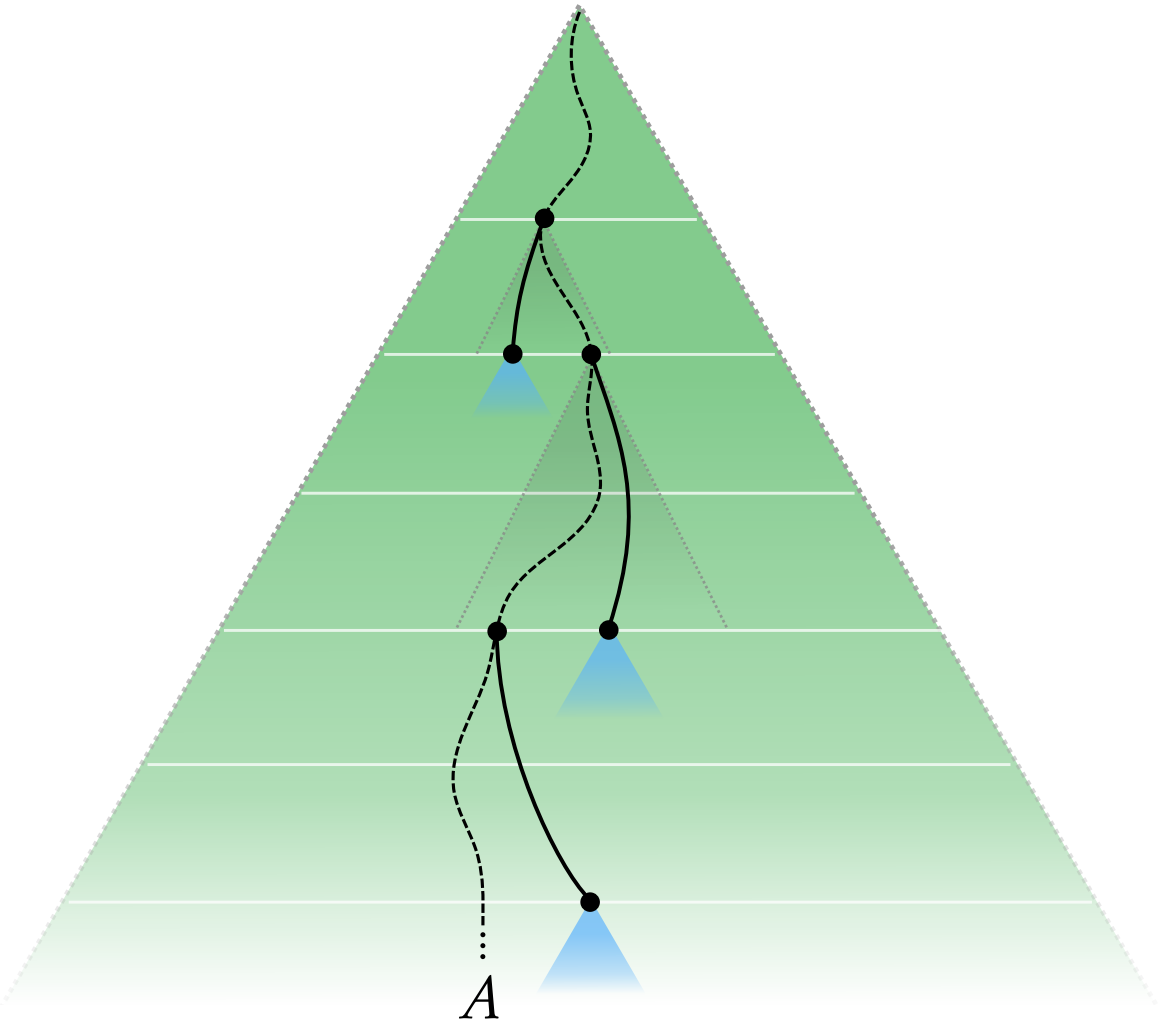
To this initial segment, another edge can be attached.

The dichotomy holds



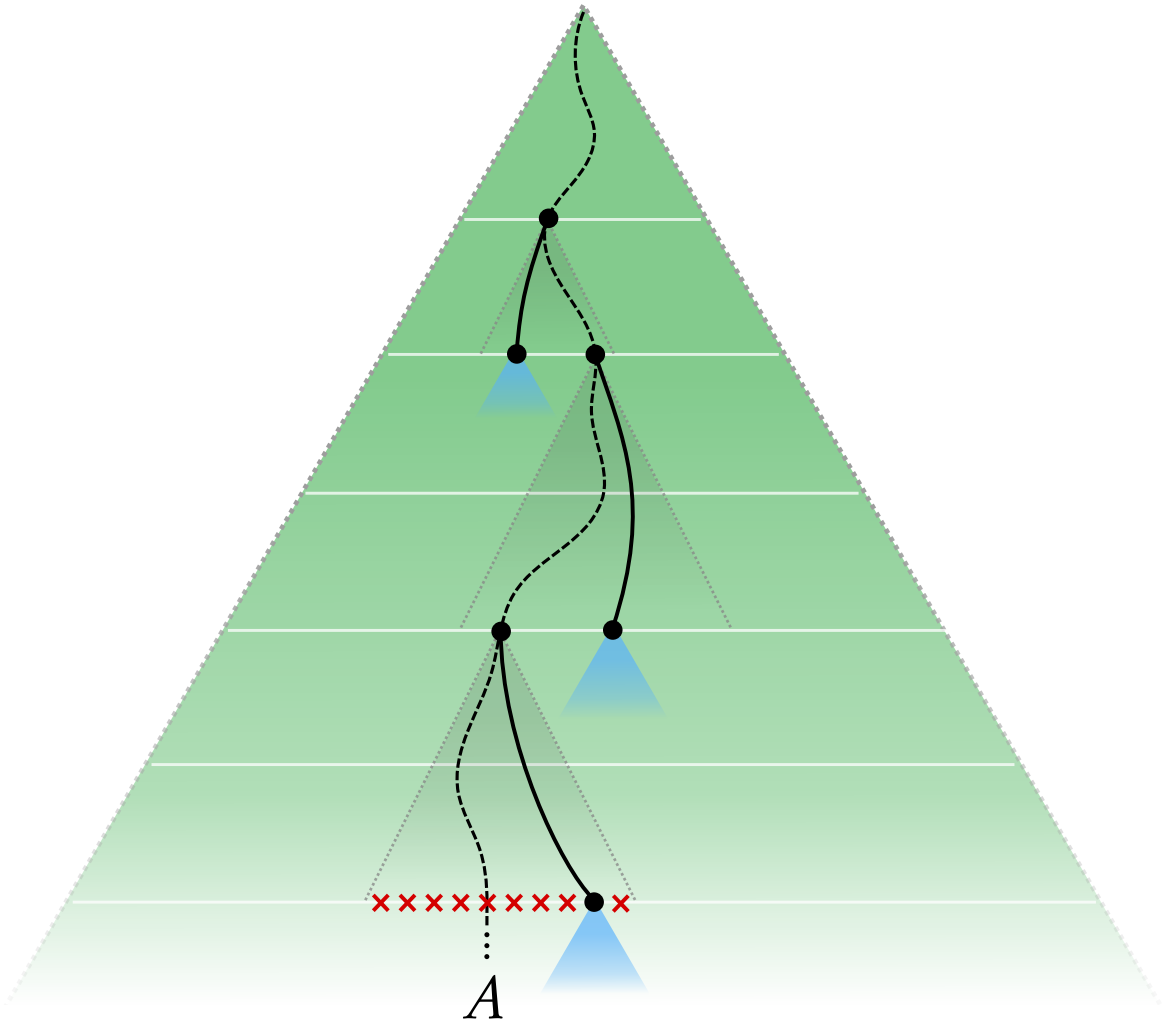
Again, a longer initial segment of A stays active.

The dichotomy holds



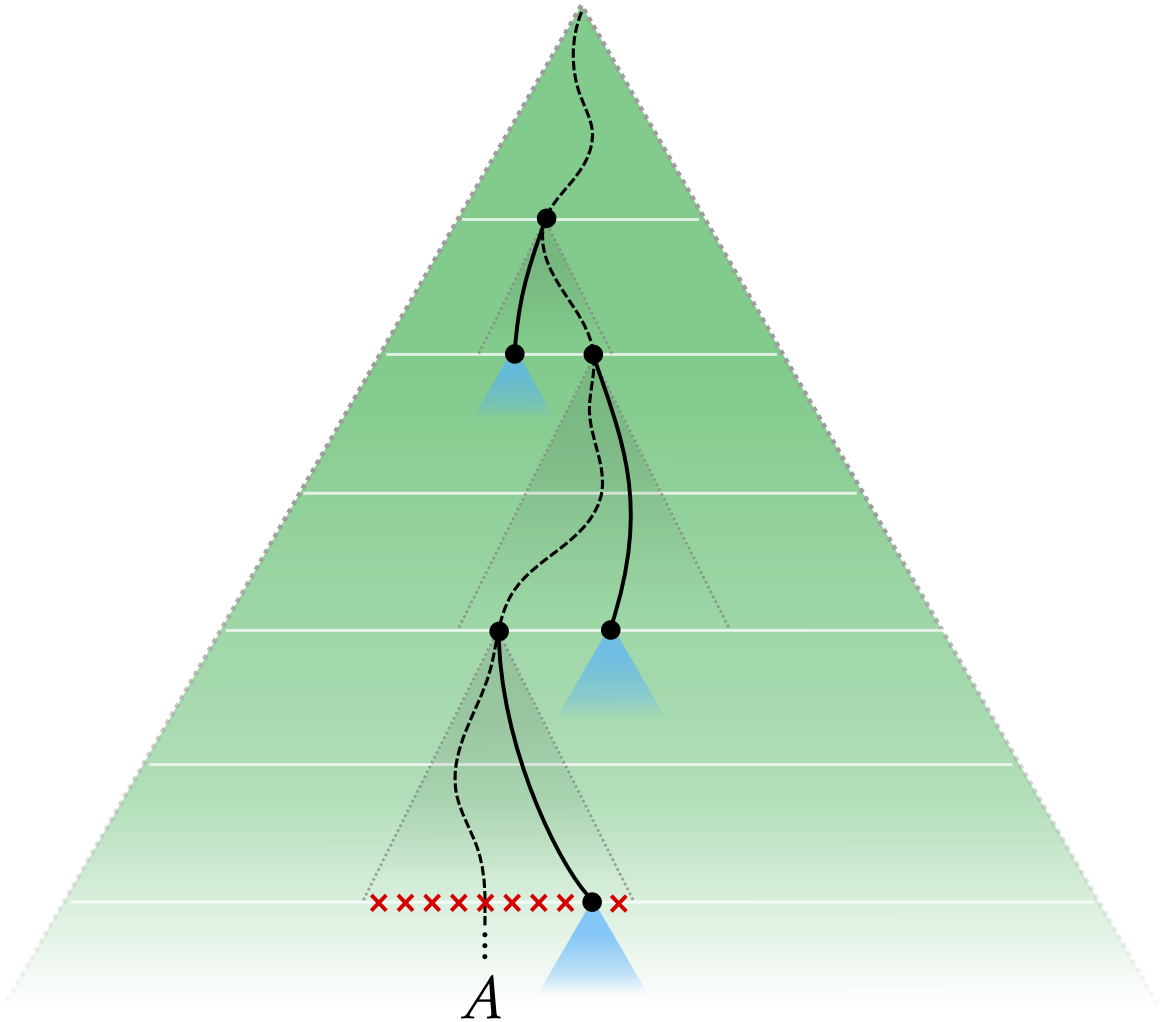
And yet another branching off edge is attached.

The dichotomy holds



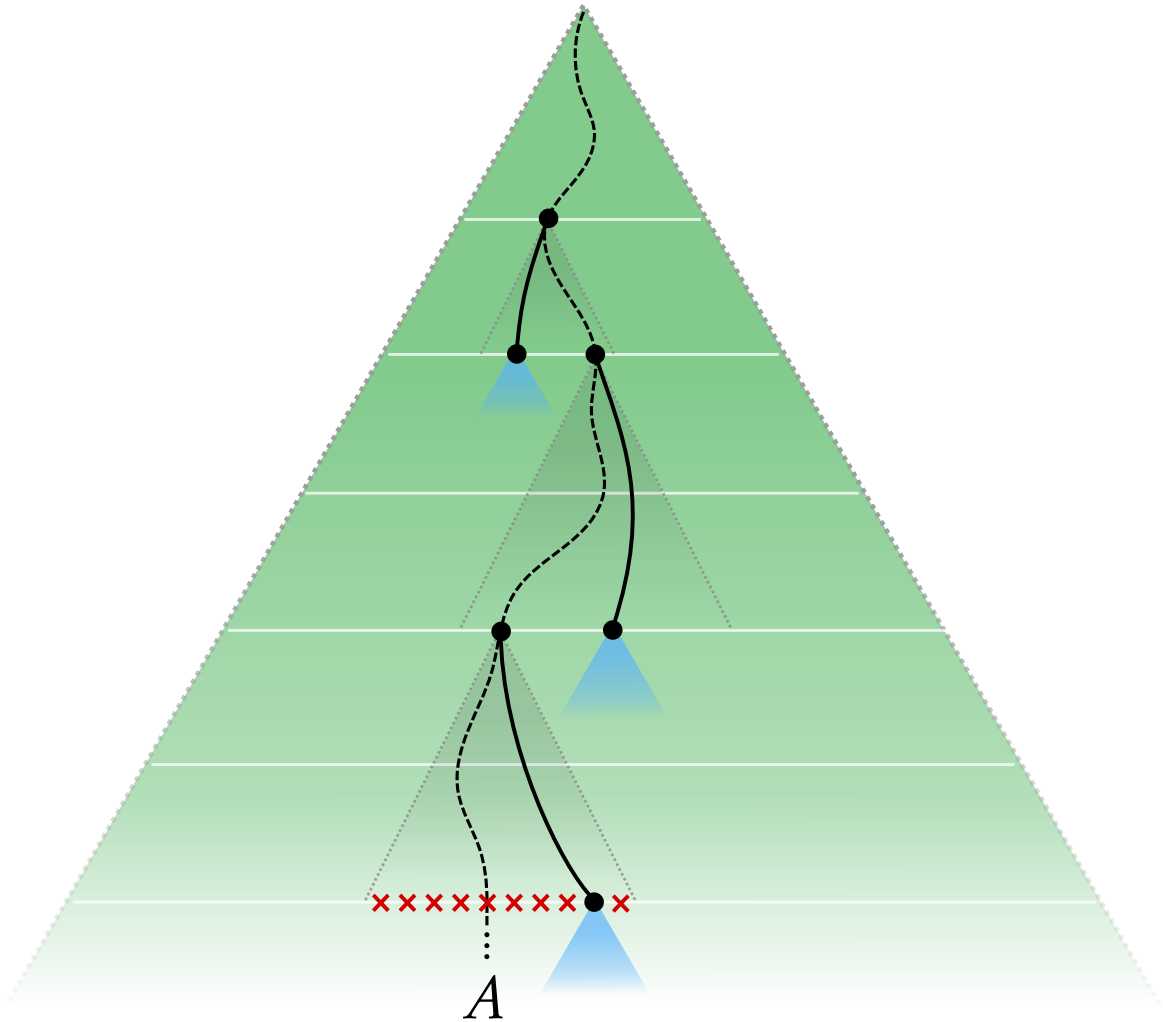
But now the countdown runs out. All flow through the other extensions is held back, and never released. A is not in the support.

The dichotomy holds



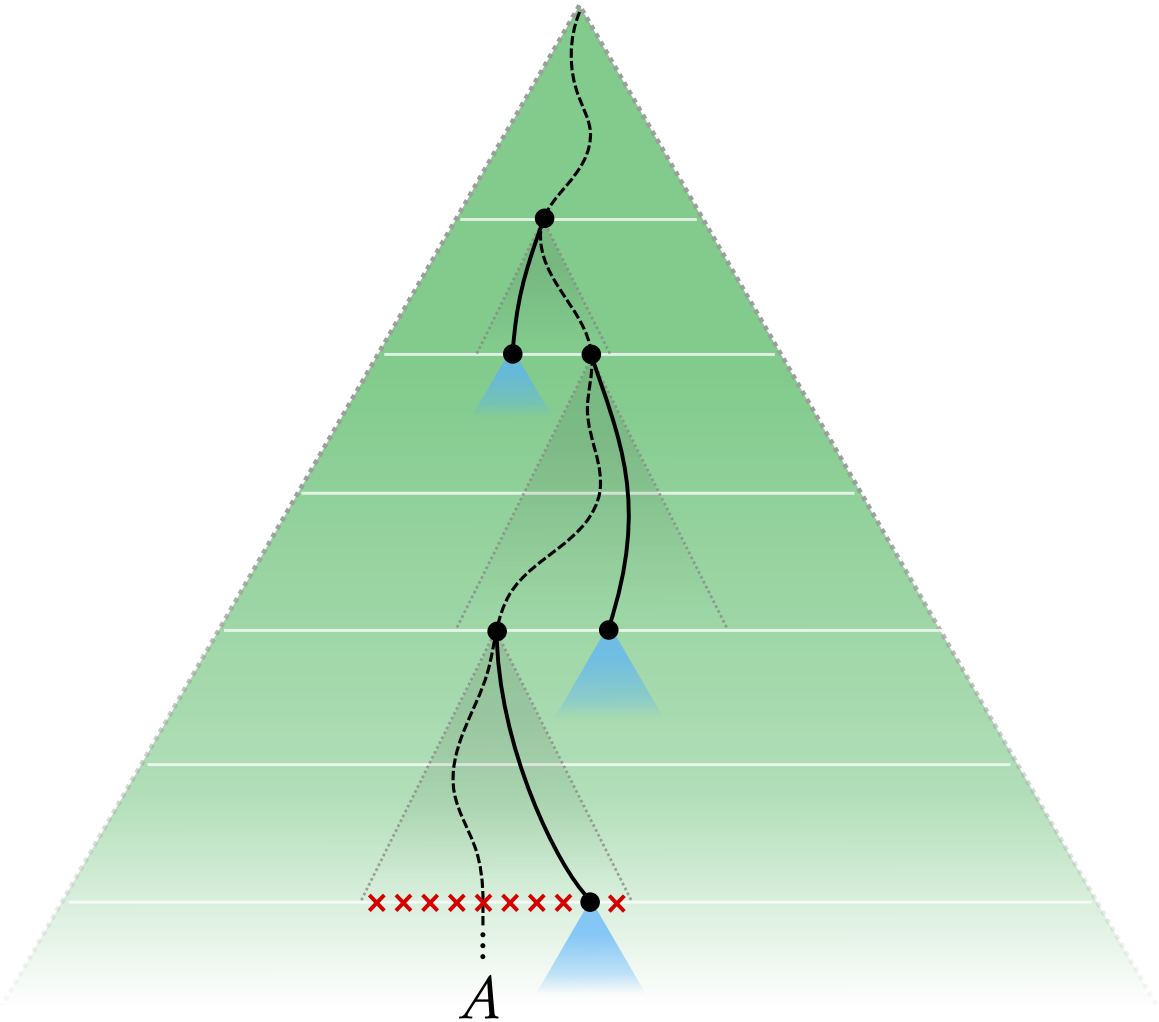
So what could go wrong with this mechanism?

The dichotomy holds



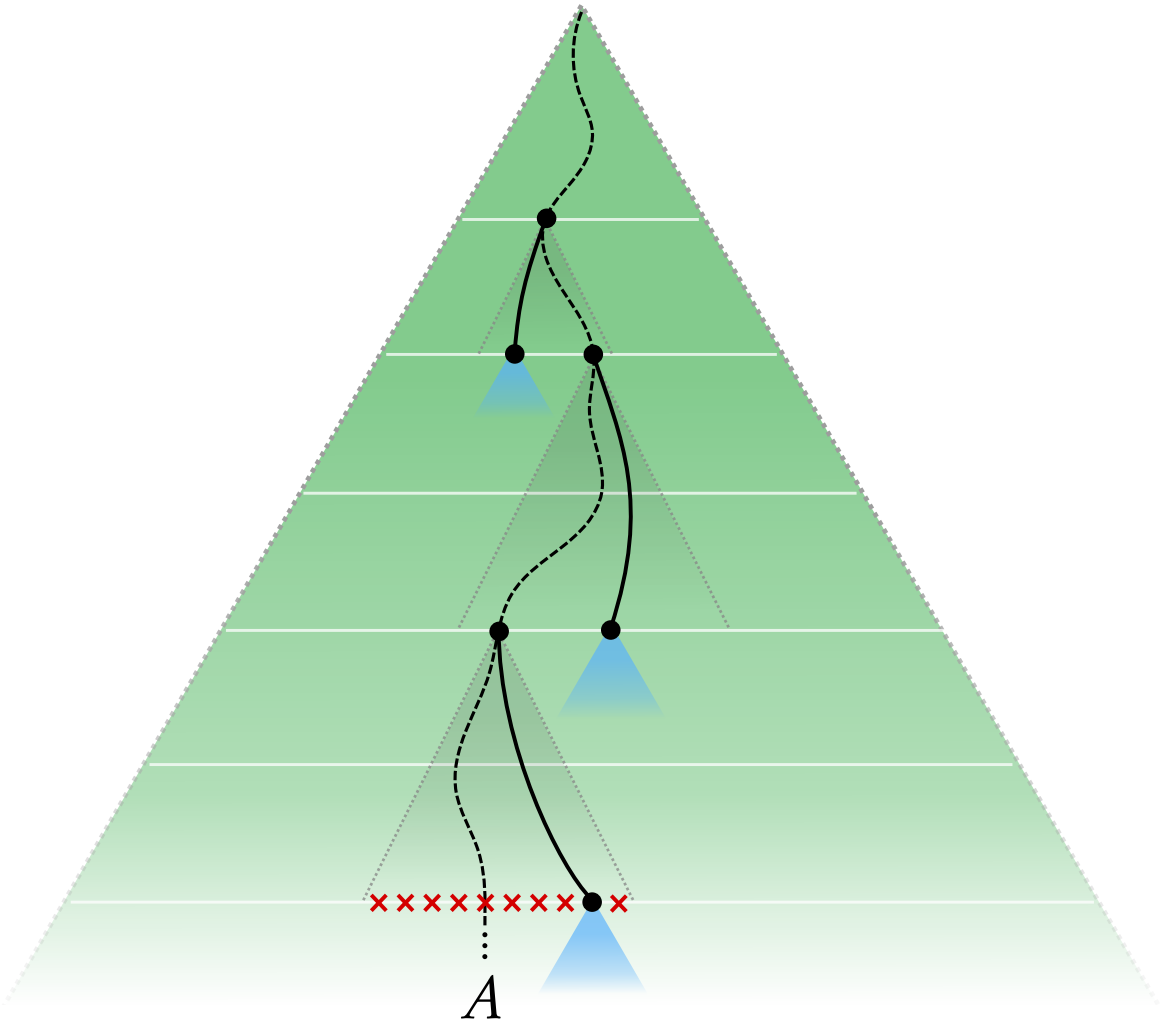
Without edges being added, there is no countdown.
A would not get removed from the support.

The dichotomy holds



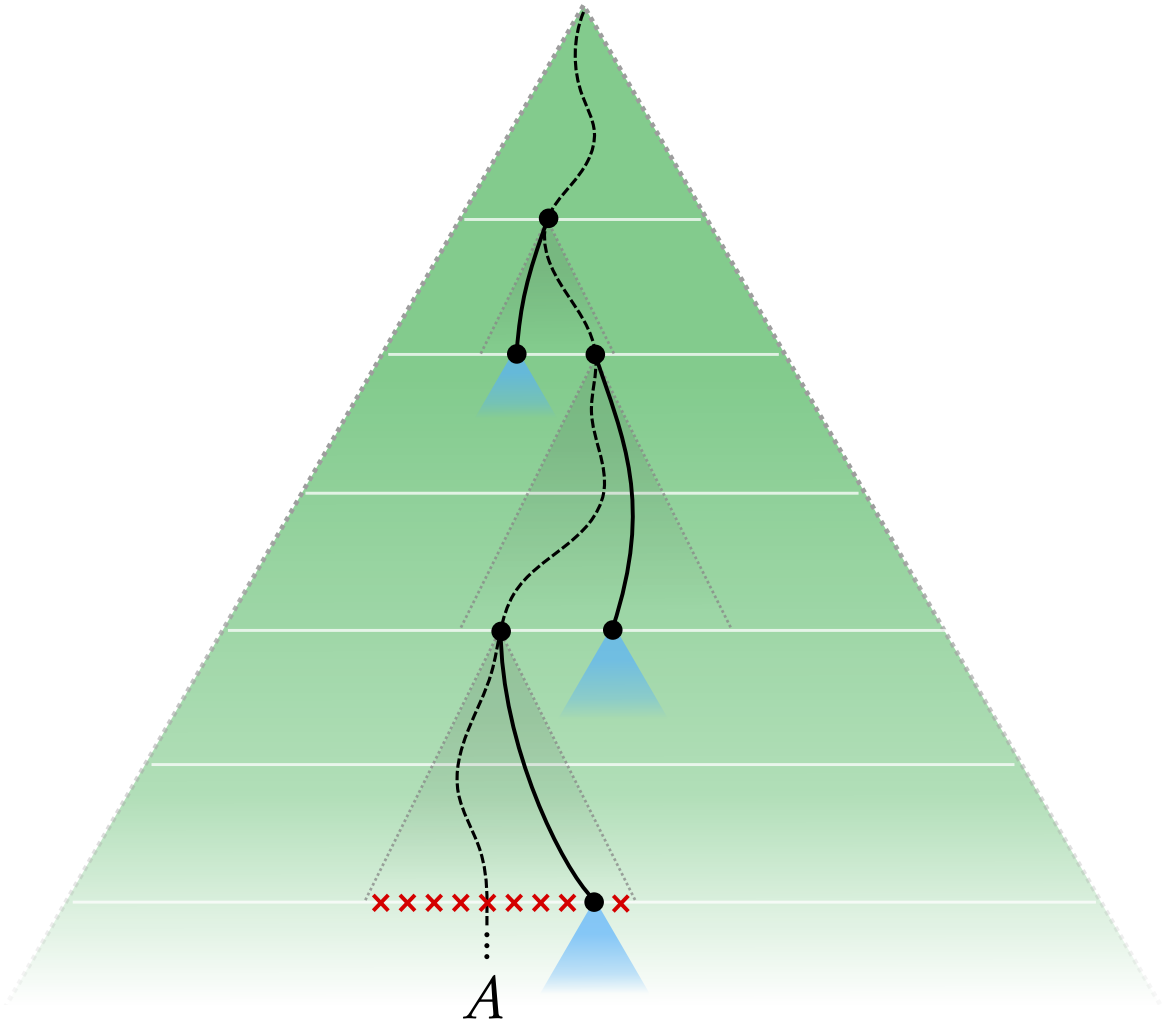
Whether the edges are added depends of the choice of B.

The dichotomy holds



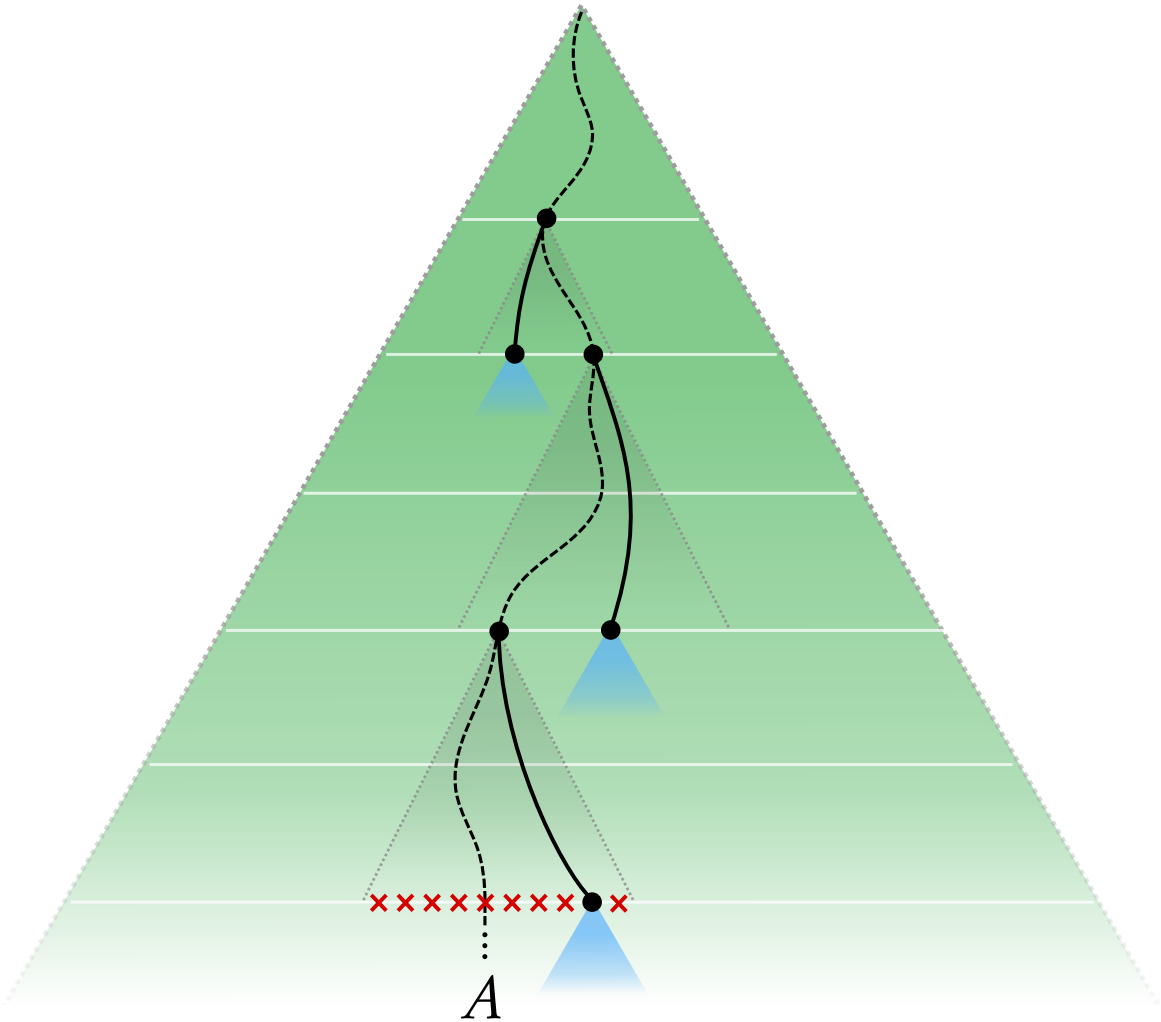
If for some reason B never wants to add edges,
the countdown cannot run out.

The dichotomy holds



So we need to show that B is such that it always permits attaching edges to longer and longer initial segments of A.

The dichotomy holds



For our particular B, this is easy to see.

The dichotomy holds

1. **Recall that we want:** For each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X) \notin \mathrm{MLR}$.
2. **Recall our choice of B:** If $\mathrm{t}(\sigma) = \mathrm{t}(\tau) = \langle j, s \rangle$ then
$$B(\sigma, \tau) \text{ holds} \iff \Phi_{j,|\tau|}^{\tau} \text{ is very long.}$$
3. **We need to show:** For $X \in \mathrm{Supp}(P)$ such that $\Phi_j(X)\!\downarrow$, for $m \in \omega$ with $\mathrm{t}(m) = \langle j, s \rangle$ where $s \in \omega$, the predicate B eventually allows an edge for task $\langle j, s \rangle$ to branch off X below $X\upharpoonright m$.
4. **But:** As $\Phi_j(X)\!\downarrow$ by assumption, for any $\ell \in \omega$ and all sufficiently large n, $|\Phi_{j,n}^{X\upharpoonright n}| \geq \ell$. Thus, $B(X\upharpoonright m, X\upharpoonright n)$ becomes true eventually.
5. So below $X\upharpoonright m$ there is always a legal candidate for an edge to be attached, and therefore some edge *will* be attached.
6. It could branch off X (thus decreasing the countdown) or it could be on X itself (because X meets the requirement). Both cases are good. □

A strengthening

1. **Theorem.** There is a left-c.e. semimeasure P such that
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X)$ does not have DNC degree.

A strengthening

1. **Theorem.** There is a left-c.e. semimeasure P such that
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X)$ does not have DNC degree.
2. Due to two theorems by Kjos-Hanssen *et al.* and Higuchi *et al.*, it suffices to prove the theorem with the third item replaced by
   - for each $X \in \mathrm{Supp}(P)$, if $\Phi(X)\!\downarrow$, then $\Phi(X)$ is not f-random for any computable function f that is unbounded along $\Phi(X)$.

A strengthening

1. **Theorem.** There is a left-c.e. semimeasure P such that
   - P has no atoms;
   - $\overline{P}(\mathrm{Supp}(P)) > 0$;
   - $\mathrm{Supp}(P)$ is a Π^0_1 class; and
   - for each $X \in \mathrm{Supp}(P)$ and each Turing functional Φ, if $\Phi(X)$ is defined, then $\Phi(X)$ does not have DNC degree.
2. Due to two theorems by Kjos-Hanssen *et al.* and Higuchi *et al.*, it suffices to prove the theorem with the third item replaced by
   - for each $X \in \mathrm{Supp}(P)$, if $\Phi(X)\!\downarrow$, then $\Phi(X)$ is not f-random for any computable function f that is unbounded along $\Phi(X)$.
3. **Proof idea.** We make B more demanding before it allows adding an edge; that is, we require functionals to produce even longer output, so that we even beat all computable functions. □

5

Infinitely many atoms

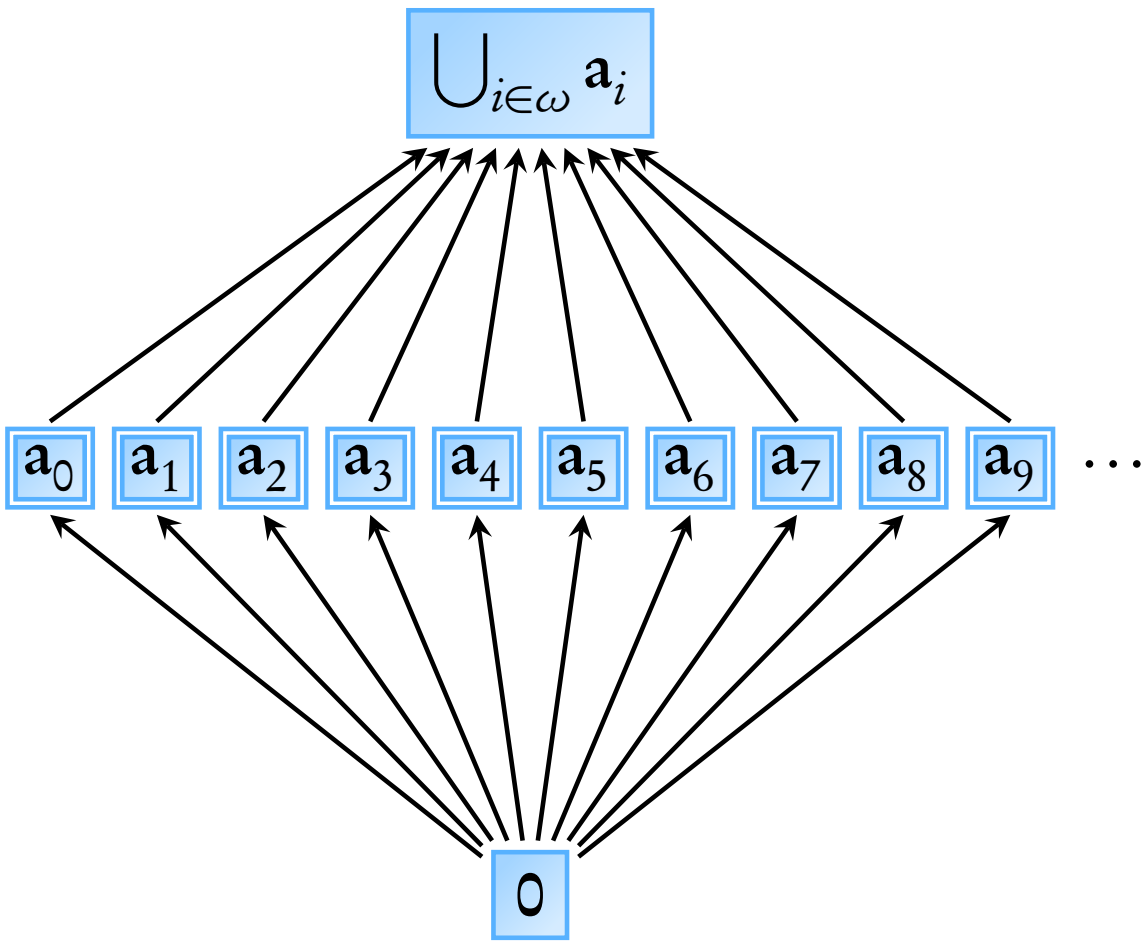
Infinitely many atoms

1. **Theorem (V'yugin).** The set of $\mathscr{D}_{\mathrm{LV}}$-atoms is countably infinite.
2. **Proof idea.**
   - V'yugin uses the discussed technique to build a single flow.
   - By picking out only the actions of some of the tasks, he derives from this single flow a countably infinite set of flows.
   - Infinitely many semimeasures are induced by these flows.
   - By having chosen the right B predicate initially, V'yugin ensured that no sequence in the support of one of these semimeasures computes a sequence in another's support.
   - Finally, the desired atoms are derived from these semimeasures.

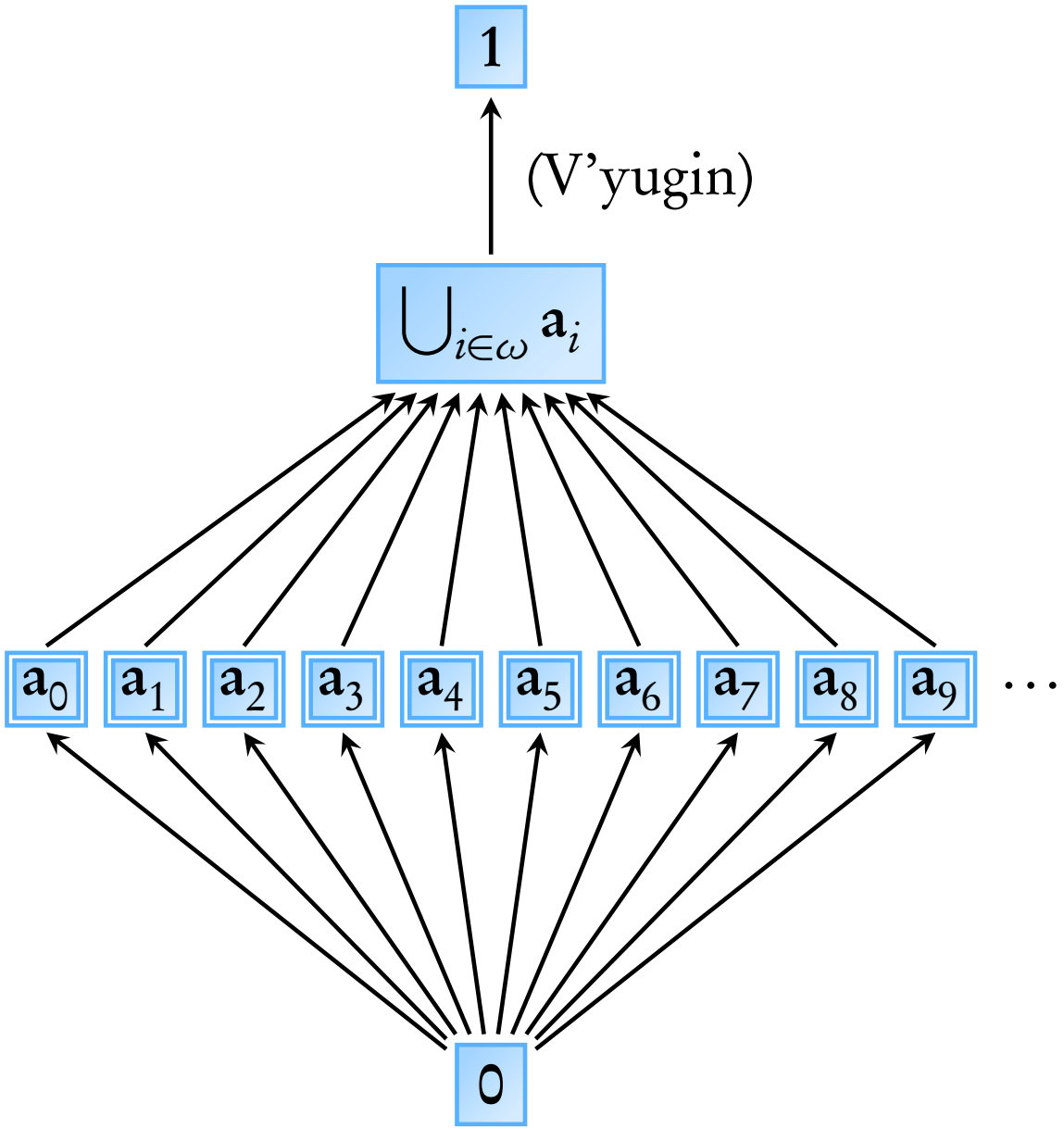
The problem with V'yugin's argument

1. To conclude that the constructed objects are indeed atoms, the semimeasures need to obey a certain zero-one law.
2. To achieve this, V'yugin modifies the construction so that the derived flows are distributed eventually identically for sequences that agree on all but finitely many bits.
3. This can result in many new extra edges in unexpected places; consequently, edges might interact in new ways with each other.
4. We can find counterexamples against the "local" Noninitial Activation Lemma; so the proof of the Measure Lemma fails.
5. The Measure Lemma itself might still hold, but any argument would have to be "global" and most likely quite involved.
6. Further problems, e.g. with the Dichotomy, look more solvable.
7. We are currently discussing possible fixes with V'yugin.

The atoms are not exhaustive



The atoms are not exhaustive



Thank you for your attention.

Examples of negligible and nonnegligible collections

1. **Proposition.** The following collections are nonnegligible:
   - the collection of sequences of DNC degree,
   - the collection of 1-generic sequences,
   - the collection of sequences of hyperimmune degree, and
   - the collection of generalized low sequences.
2. **Proposition.** The following collections are negligible:
   - the collection of sequences of PA degree,
   - the collection of sequences of high degree,
   - the collection of 2-generic sequences, and
   - the collection of noncomputable, hyperimmune-free sequences.