

One Part of Shannon's Source Coding Theorem

Quentin Hibon

February 6, 2026

Abstract

This document contains a proof of the necessary condition on the code rate of a source code, namely that this code rate is bounded by the entropy of the source. This represents one half of Shannon's source coding theorem, which is itself an equivalence.

This proof is taken directly from the textbook [1], and transcribed rather literally into Isabelle. It is thus easier to keep the textbook proof in mind to understand this formal proof.

Contents

1	Basic types	1
2	Locale for the source coding theorem	2
3	Source coding theorem, direct: the entropy is a lower bound of the code rate	2
3.1	The letter set	2
3.2	Codes and words	2
3.3	Related to the Kraft theorem	4
3.4	Inequality of the kraft sum (source coding theorem, direct)	6
3.4.1	Sum manipulation lemmas and McMillan theorem	6
3.4.2	Technical lemmas about the logarithm	10
3.4.3	KL divergence and properties	10

```
theory Source-Coding-Theorem  
imports HOL-Probability.Information  
begin
```

1 Basic types

```
type-synonym bit = bool  
type-synonym bword = bit list  
type-synonym letter = nat
```

type-synonym 'b word = 'b list

type-synonym 'b encoder = 'b word \Rightarrow bword

type-synonym 'b decoder = bword \Rightarrow 'b word option

2 Locale for the source coding theorem

locale source-code = information-space +

fixes fi :: 'b \Rightarrow real

fixes X :: 'a \Rightarrow 'b

assumes distr-i: simple-distributed M X fi

assumes b-val: b = 2

fixes enc::'b encoder

fixes dec::'b decoder

assumes real-code:

dec (enc x) = Some x

enc w = [] \longleftrightarrow w = []

x \neq [] \longrightarrow enc x = enc [hd x] @ enc (tl x)

3 Source coding theorem, direct: the entropy is a lower bound of the code rate

context source-code

begin

3.1 The letter set

definition L :: 'b set **where**

L \equiv X ' space M

lemma fin-L: finite L

using L-def distr-i

by auto

lemma emp-L: L \neq {}

using L-def subprob-not-empty

by auto

3.2 Codes and words

abbreviation real-word :: 'b word \Rightarrow bool **where**

real-word w \equiv (set w \subseteq L)

abbreviation k-words :: nat \Rightarrow ('b word) set **where**

k-words k \equiv {w. length w = k \wedge real-word w}

lemma *rw-tail*:

assumes *real-word* w

shows $w = [] \vee \text{real-word } (\text{tl } w)$

by (*meson* *assms* *list.set-sel*(2) *subset-code*(1))

definition *code-word-length* :: 'e *encoder* \Rightarrow 'e \Rightarrow *nat* **where**

code-word-length e $l = \text{length } (e [l])$

abbreviation *cw-len* :: 'b \Rightarrow *nat* **where**

cw-len $l \equiv \text{code-word-length } \text{enc } l$

definition *code-rate* :: 'e *encoder* \Rightarrow ('a \Rightarrow 'e) \Rightarrow *real* **where**

code-rate e $Xo = \text{expectation } (\lambda a. (\text{code-word-length } e ((Xo) a)))$

lemma *fi-pos*: $i \in L \Longrightarrow 0 \leq \text{fi } i$

using *simple-distributed-nonneg*[*OF* *distr-i*] *L-def* **by** *auto*

lemma (*in prob-space*) *simp-exp-composed*:

assumes X : *simple-distributed* M X Px

shows $\text{expectation } (\lambda a. f (X a)) = (\sum x \in X \text{'space } M. f x * Px x)$

using *distributed-integral*[*OF* *simple-distributed*[*OF* X], *of* f]

simple-distributed-nonneg[*OF* X]

lebesgue-integral-count-space-finite[*OF* *simple-distributed-finite*[*OF* X], *of* $\lambda x. f$
 $x * Px x$]

by (*simp* *add*: *ac-simps*)

lemma *cr-rw*:

code-rate *enc* $X = (\sum i \in X \text{'space } M. \text{fi } i * \text{cw-len } i)$

using *simp-exp-composed*[*OF* *distr-i*, *of* *cw-len*]

by (*simp* *add*: *mult commute* *code-rate-def*)

abbreviation *cw-len-concat* :: 'b *word* \Rightarrow *nat* **where**

cw-len-concat $w \equiv \text{foldr } (\lambda x s. (\text{cw-len } x) + s) w 0$

lemma *cw-len-length*: *cw-len-concat* $w = \text{length } (\text{enc } w)$

proof (*induction* w)

case *Nil*

show *?case* **using** *real-code* **by** *simp*

case (*Cons* a w)

have *cw-len-concat* ($a \# w$) = *cw-len* a + *cw-len-concat* w **by** *simp*

thus *?case* **using** *code-word-length-def* *real-code* *Cons*

by (*metis* *length-append* *list.distinct*(1) *list.sel*(1) *list.sel*(3))

qed

lemma *maj-fold*:

assumes $\bigwedge l. l \in L \Longrightarrow f l \leq \text{bound}$

assumes *real-word* w

shows $\text{foldr } (\lambda x s. f x + s) w 0 \leq \text{length } w * \text{bound}$

using *assms*

by(*induction w*) (*simp,fastforce*)

definition *max-len* :: nat **where**
max-len = Max (($\lambda x. cw-len\ x$) ‘ L)

lemma *max-cw*:
 $l \in L \implies cw-len\ l \leq max-len$
 by (*simp add: max-len-def fin-L*)

3.3 Related to the Kraft theorem

definition \mathcal{K} :: real **where**
 $\mathcal{K} = (\sum_{i \in L. 1 / b \wedge (cw-len\ i)})$

lemma *pos-cw-len*: $0 < 1 / b \wedge cw-len\ i$ **using** *b-gt-1* **by** *simp*

lemma *K-pos*: $0 < \mathcal{K}$
using *emp-L fin-L pos-cw-len sum-pos K-def*
by *metis*

lemma *K-pow*: $\mathcal{K} = (\sum_{i \in L. 1 / b^{powr\ cw-len\ i})$
using *powr-realpow b-gt-1*
by (*simp add: K-def*)

lemma *k-words-rel*:
 $k-words\ (Suc\ k) = \{w. (hd\ w \in L \wedge tl\ w \in k-words\ k \wedge w \neq [])\}$
proof
fix *k*
show $k-words\ (Suc\ k) \subseteq \{w. (hd\ w \in L \wedge tl\ w \in k-words\ k \wedge w \neq [])\}$ (**is** *?l*
 \subseteq *?r*)
proof
fix *w*
assume *w-kw*: $w \in k-words\ (Suc\ k)$
hence *real-word w* **by** *simp*
hence $hd\ w \in L$
by (*metis (mono-tags) w-kw hd-in-set list.size(3) mem-Collect-eq nat.distinct(1) subset-code(1)*)
moreover **have** $length\ w = Suc\ k$ **using** *w-kw* **by** *simp*
moreover **hence** $w \neq []$ **by** *auto*
moreover **have** *real-word (tl w)* **using** $\langle real-word\ w \rangle$ *calculation(3) rw-tail*
by *auto*
ultimately **show** $w \in ?r$ **using** *w-kw* **by** *simp*
qed
next
fix *k*
show $k-words\ (Suc\ k) \supseteq \{w. (hd\ w \in L \wedge tl\ w \in k-words\ k \wedge w \neq [])\}$
proof
fix *w*
assume *asm*: $w \in \{w. hd\ w \in L \wedge tl\ w \in \{w. length\ w = k \wedge real-word\ w\}$

$\wedge w \neq []$
hence $hd\ w \in L \wedge length\ (tl\ w) = k \wedge real\text{-}word\ (tl\ w)$ **by** *simp*
hence *real-word w*
by (*metis empty-iff insert-subset list.collapse list.set(1) set-simps(2) subsetI*)
moreover hence $length\ w = Suc\ k$ **using** *asm* **by** *auto*
ultimately show $w \in k\text{-}words\ (Suc\ k)$ **by** *simp*
qed
qed

lemma *bij-k-words*:
shows *bij-betw* $(\lambda wi. Cons\ (fst\ wi)\ (snd\ wi))\ (L \times k\text{-}words\ k)\ (k\text{-}words\ (Suc\ k))$
unfolding *bij-betw-def*
proof
fix k
let $?f = (\lambda wi. Cons\ (fst\ wi)\ (snd\ wi))$
let $?S = L \times (k\text{-}words\ k)$
let $?T = k\text{-}words\ (Suc\ k)$
show *inj-on* $?f\ ?S$ **by** (*simp add: inj-on-def*)
show $?f' ?S = ?T$
proof (*rule ccontr*)
assume $?f' ?S \neq ?T$
hence $\exists w. w \in ?T \wedge w \notin ?f' ?S$ **by** *auto*
then obtain w **where** *asm*: $w \in ?T \wedge w \notin ?f' ?S$ **by** *blast*
hence $w = ?f\ (hd\ w, tl\ w)$ **using** *k-words-rel* **by** *simp*
moreover have $(hd\ w, tl\ w) \in ?S$ **using** *k-words-rel asm* **by** *simp*
ultimately have $w \in ?f' ?S$ **by** *blast*
thus *False* **using** *asm* **by** *simp*
qed
qed

lemma *finite-k-words: finite* $(k\text{-}words\ k)$
proof (*induct k*)
case 0
show *?case* **by** *simp*
case $(Suc\ n)$
thus *?case* **using** *bij-k-words bij-betw-finite fin-L* **by** *blast*
qed

lemma *cartesian-product*:
fixes $f::('c \Rightarrow real)$
fixes $g::('d \Rightarrow real)$
assumes *finite A*
assumes *finite B*
shows $(\sum b \in B. g\ b) * (\sum a \in A. f\ a) = (\sum ab \in A \times B. f\ (fst\ ab) * g\ (snd\ ab))$
using *bilinear-times bilinear-sum* [**where** $h = (\lambda x\ y. x * y)$ **and** $f = f$ **and** $g = g$]
assms
by (*metis (erased, lifting) sum.cong split-beta' Groups.ab-semigroup-mult-class.mult commute*)

lemma *K-power*:

shows $\mathcal{K} \hat{\ }^k = (\sum w \in (k\text{-words } k). 1 / b^{\wedge}(cw\text{-len-concat } w))$
proof (*induct k*)
 case 0
 have $k\text{-words } 0 = \{\}\}$ **by** *auto*
 thus *?case* **by** *simp*
next
 case (*Suc n*)
 have $\mathcal{K} \hat{\ }^{Suc\ n} = \mathcal{K} \hat{\ }^n * \mathcal{K}$ **by** *simp*
 also have $\dots = (\sum w \in k\text{-words } n. 1 / b^{\wedge}cw\text{-len-concat } w) * (\sum i \in L. 1 / b^{\wedge}cw\text{-len } i)$
 using *Suc.hyps* $\mathcal{K}\text{-def}$ **by** *auto*
 also have $\dots = (\sum wi \in L \times k\text{-words } n. 1 / b^{\wedge}cw\text{-len } (fst\ wi)) * (1 / b^{\wedge}cw\text{-len-concat } (snd\ wi))$
 using *fin-L finite-k-words cartesian-product*
 by *blast*
 also have $\dots = (\sum wi \in L \times k\text{-words } n. 1 / b^{\wedge}(cw\text{-len-concat } (snd\ wi) + cw\text{-len } (fst\ wi)))$
 by (*metis* (*no-types, lifting*) *power-add add.commute power-one-over*)
 also have $\dots = (\sum wi \in L \times k\text{-words } n. 1 / b^{\wedge}cw\text{-len-concat } (fst\ wi \# snd\ wi))$
 by (*metis* (*erased, lifting*) *add.commute comp-apply foldr.simps(2)*)
 also have $\dots = (\sum w \in (k\text{-words } (Suc\ n)). 1 / b^{\wedge}(cw\text{-len-concat } w))$
 using *sum.reindex-bij-betw [OF bij-k-words]* **by** *fastforce*
 finally show *?case* **by** *simp*
qed

lemma *bound-len-concat:*

shows $w \in k\text{-words } k \implies cw\text{-len-concat } w \leq k * max\text{-len}$
using *max-cw maj-fold* **by** *blast*

3.4 Inequality of the kraft sum (source coding theorem, direct)

3.4.1 Sum manipulation lemmas and McMillan theorem

lemma *sum-vimage-proof:*

fixes $g::nat \Rightarrow real$
assumes $\bigwedge w. f\ w < bd$
shows $finite\ S \implies (\sum w \in S. g\ (f\ w)) = (\sum m=0..<bd. (card\ ((f^{-1}\{m\}) \cap S)) * g\ m)$
(is $- \implies - = (\sum m=0..<bd. ?ff\ m\ S)$ **)**
proof (*induct S rule: finite-induct*)
 case *empty*
 show *?case* **by** *simp*
next
 case (*insert x F*)
 let $?rr = (\sum m = 0..<bd. ?ff\ m\ (insert\ x\ F))$
 have $(f\ x) \in \{0..<bd\}$ **using** *assms* **by** *simp*
 hence $\bigwedge h::(nat \Rightarrow real). (\sum m=0..<bd. h\ m) = (\sum y \in (\{0..<bd\} - \{f\ x\}). h\ y) + h\ (f\ x)$

by (metis diff-add-cancel finite-atLeastLessThan sum-diff1)
moreover hence
 $(\sum m = 0..<bd. ?ff\ m\ (insert\ x\ F))$
 $= (\sum m \in \{0..<bd\} - \{f\ x\}. ?ff\ m\ (insert\ x\ F)) + card\ (f - \{f\ x\} \cap F) * g\ (f\ x) + g\ (f\ x)$
 by (simp add: semiring-normalization-rules(2), simp add: insert)
ultimately have $(\sum m = 0..<bd. ?ff\ m\ (insert\ x\ F)) = (\sum m \in \{0..<bd\}. ?ff\ m\ F) + g\ (f\ x)$
 by fastforce
thus ?case using insert by simp
qed

lemma sum-vimage:

fixes $g :: nat \Rightarrow real$

assumes *bounded*: $\bigwedge w. w \in S \implies f\ w < bd$ **and** $0 < bd$

assumes *finite*: *finite* S

shows $(\sum w \in S. g\ (f\ w)) = (\sum m = 0..<bd. (card\ ((f - \{m\}) \cap S)) * g\ m)$

(**is** $?s1 = ?s2$)

proof –

let $?ff = (\lambda x. if\ x \in S\ then\ f\ x\ else\ 0)$

let $?ss1 = (\sum w \in S. g\ (?ff\ w))$

let $?ss2 = (\sum m = 0..<bd. (card\ ((?ff - \{m\}) \cap S)) * g\ m)$

have $?s1 = ?ss1$ **by** *simp*

moreover have $\bigwedge m. ?ff - \{m\} \cap S = f - \{m\} \cap S$ **by** *auto*

moreover hence $?s2 = ?ss2$ **by** *simp*

moreover have $\bigwedge w. ?ff\ w < bd$ **using** *assms* **by** *simp*

moreover hence $?ss1 = ?ss2$ **using** *sum-vimage-proof*[of $?ff$] *finite* **by** *blast*

ultimately show $?s1 = ?s2$ **by** *metis*

qed

lemma \mathcal{K} -*rw*:

$(\sum w \in (k\text{-words}\ k). 1 / b^{(cw\text{-len}\text{-concat}\ w)}) = (\sum m = 0..<Suc\ (k * max\text{-len}).$

$card\ (k\text{-words}\ k \cap$

$((cw\text{-len}\text{-concat}) - \{m\})) * (1 / b^m))$ (**is** $?L = ?R$)

proof –

have $\bigwedge w. w \in k\text{-words}\ k \implies cw\text{-len}\text{-concat}\ w < Suc\ (k * max\text{-len})$

by (simp add: bound-len-concat le-imp-less-Suc)

moreover have

$?R = (\sum m = 0..<Suc\ (k * max\text{-len}).$

$(card\ (cw\text{-len}\text{-concat} - \{m\} \cap k\text{-words}\ k)) * (1 / b^m))$

by (metis Int-commute)

moreover have $0 < Suc\ (k * max\text{-len})$ **by** *simp*

ultimately show *?thesis*

using *finite-k-words*

sum-vimage[**where** $f = cw\text{-len}\text{-concat}$ **and** $g = \lambda i. 1 / (b^i)$]

by *fastforce*

qed

definition *set-of-k-words-length-m* :: $nat \Rightarrow nat \Rightarrow 'b$ word set **where**

$set\text{-of-}k\text{-words-length-}m\ k\ m = \{xk. xk \in k\text{-words } k\} \cap (cw\text{-len-concat})\text{-}\{m\}$

lemma *am-inj-code*: *inj-on enc ((cw-len-concat)-{m}) (is inj-on - ?s)*
using *inj-on-def[of enc ?s] real-code*
by (*metis option.inject*)

lemma *img-inc*: *enc{cw-len-concat}-{m} \subseteq {bl. length bl = m} using cw-len-length*
by *auto*

lemma *bool-lists-card*: *card {bl::bool list. length bl = m} = b^m*
using *card-lists-length-eq[of UNIV::bool set]*
by (*simp add: b-val*)

lemma *bool-list-fin*: *finite {bl::bool list. length bl = m}*
using *finite-lists-length-eq[of UNIV::bool set]*
by (*simp add: b-val*)

lemma *set-of-k-words-bound*:

shows *card (set-of-k-words-length-m k m) \leq b^m (is ?c \leq ?b)*

proof –

have *card-w-len-m-bound*: *card (cw-len-concat)-{m} \leq b^m*

by (*metis (no-types, lifting) am-inj-code bool-list-fin bool-lists-card card-image card-mono*

img-inc of-nat-le-iff)

have *set-of-k-words-length-m k m \subseteq (cw-len-concat)-{m}*

by (*simp add: set-of-k-words-length-m-def*)

hence *card (set-of-k-words-length-m k m) \leq card ((cw-len-concat)-{m})*

by (*metis (no-types, lifting) am-inj-code bool-list-fin card.infinite card-0-eq card-image card-mono empty-iff finite-subset img-inc inf-img-fin-dom*)

thus *?thesis using card-w-len-m-bound by simp*

qed

lemma *empty-set-k-words*:

assumes *0 < k*

shows *set-of-k-words-length-m k 0 = {}*

proof(*rule ccontr*)

assume \neg *set-of-k-words-length-m k 0 = {}*

hence $\exists x. x \in set\text{-of-}k\text{-words-length-}m\ k\ 0$ **by** *auto*

then obtain *x* **where** *x-def*: *x \in set-of-k-words-length-m k 0* **by** *auto*

hence *x \neq [] unfolding set-of-k-words-length-m-def using assms* **by** *auto*

moreover have *cw-len-concat (hd x#tl x) = cw-len-concat (tl x) + cw-len (hd x)*

by (*metis add.commute comp-apply foldr.simps(2)*)

moreover have *enc [(hd x)] \neq [] using assms real-code* **by** *blast*

moreover hence *0 < cw-len (hd x) unfolding code-word-length-def* **by** *simp*

ultimately have *x \notin set-of-k-words-length-m k 0* **by** (*simp add:set-of-k-words-length-m-def*)

thus *False using x-def* **by** *simp*

qed

lemma \mathcal{K} -rw2:
assumes $0 < k$
shows $(\sum m=0..<Suc (k * max-len). card (set-of-k-words-length-m k m) / b^m) \leq (k * max-len)$
proof –
have
 $(\sum m=1..<Suc (k * max-len). card (set-of-k-words-length-m k m) / b^m)$
 $\leq (\sum m=1..<Suc(k * max-len). b^m / b^m)$
using *set-of-k-words-bound b-val*
Groups-Big.sum-mono[of $\{1..<Suc(k * max-len)\}$]
 $(\lambda m. (card (set-of-k-words-length-m k m) / b^m) \lambda m. b^m / b^m)$
by *simp*
moreover **have** $(\sum m=1..<Suc(k * max-len). b^m / b^m) = (\sum m=1..<Suc(k * max-len). 1)$
using *b-gt-1* **by** *simp*
moreover **have** $(\sum m=1..<Suc(k * max-len). 1) = (k * max-len)$
by *simp*
ultimately **have**
 $(\sum m = 1..<Suc (k * max-len). card (set-of-k-words-length-m k m) / b^m)$
 $\leq k * max-len$
by (*metis One-nat-def card-atLeastLessThan card-eq-sum diff-Suc-Suc real-of-card*)
thus *?thesis* **using** *empty-set-k-words assms*
by (*simp add: sum-shift-lb-Suc0-0-upt split: if-split-asm*)
qed

lemma \mathcal{K} -power-bound :
assumes $0 < k$
shows $\mathcal{K}^k \leq k * max-len$
using *assms* \mathcal{K} -power \mathcal{K} -rw \mathcal{K} -rw2
by (*simp add: set-of-k-words-length-m-def*)

theorem *McMillan* :
shows $\mathcal{K} \leq 1$
proof –
have *ineq*: $\bigwedge k. 0 < k \implies \mathcal{K} \leq \text{root } k \ k * \text{root } k \ max-len$
using \mathcal{K} -pos \mathcal{K} -power-bound
by (*metis (no-types, opaque-lifting) not-less-of-nat-0-le-iff of-nat-mult power-strict-mono real-root-mult real-root-pos-pos-le real-root-pos-unique real-root-power*)
hence $0 < max-len \implies (\lambda k. \text{root } k \ k * \text{root } k \ max-len) \longrightarrow 1$
by (*auto intro!: tendsto-eq-intros LIMSEQ-root LIMSEQ-root-const*)
moreover **have** $\forall n \geq 1. \mathcal{K} \leq \text{root } n \ n * \text{root } n \ max-len$
using *ineq* **by** *simp*
moreover **have** $max-len = 0 \implies \mathcal{K} \leq 1$ **using** *ineq* **by** *fastforce*
ultimately **show** $\mathcal{K} \leq 1$ **using** *LIMSEQ-le-const* **by** *blast*
qed

lemma *entropy-rw*: $\mathcal{H}(X) = -(\sum i \in L. fi \ i * \log b (fi \ i))$
using *entropy-simple-distributed*[*OF distr-i*]
by (*simp add: L-def*)

3.4.2 Technical lemmas about the logarithm

lemma *log-mult-ext3*:

$0 \leq x \implies 0 < y \implies 0 < z \implies x * \log b (x*y*z) = x * \log b (x*y) + x * \log b z$

by (*metis dual-order.irrefl log-mult ring-distrib(1) mult-eq-0-iff*)

lemma *log-mult-ext2*:

$0 \leq x \implies 0 < y \implies x * \log b (x*y) = x * \log b x + x * \log b y$

using *log-mult-ext3*[**where** $y=1$] **by** *simp*

3.4.3 KL divergence and properties

definition *KL-div* $:: 'b \text{ set} \Rightarrow ('b \Rightarrow \text{real}) \Rightarrow ('b \Rightarrow \text{real}) \Rightarrow \text{real}$ **where**

$KL\text{-div } S \ a \ d = (\sum i \in S. a \ i * \log b (a \ i / d \ i))$

lemma *KL-div-mul*:

assumes $0 < d \ d \leq 1$

assumes $\bigwedge i. i \in S \implies 0 \leq a \ i$

assumes $\bigwedge i. i \in S \implies 0 < e \ i$

shows $KL\text{-div } S \ a \ e \geq KL\text{-div } S \ a \ (\lambda i. e \ i / d)$

unfolding *KL-div-def*

proof –

{

fix i

assume $i \in S$

hence $a \ i / (e \ i / d) \leq a \ i / e \ i$ **using** *assms*

by (*metis (no-types) div-by-1 frac-le less-imp-triv not-less*)

hence $\log b (a \ i / (e \ i / d)) \leq \log b (a \ i / e \ i)$ **using** *assms*

by (*smt (verit, best) Transcendental.log-mono <i ∈ S> b-gt-1 diff-divide-distrib divide-pos-pos*)

}

thus $(\sum i \in S. a \ i * \log b (a \ i / (e \ i / d))) \leq (\sum i \in S. a \ i * \log b (a \ i / e \ i))$

by (*meson mult-left-mono assms sum-mono*)

qed

lemma *KL-div-pos*:

fixes $a \ e :: 'b \Rightarrow \text{real}$

assumes *fin*: $\text{finite } S$

assumes *nemp*: $S \neq \{\}$

assumes *non-null*: $\bigwedge i. i \in S \implies 0 < a \ i \ \bigwedge i. i \in S \implies 0 < e \ i$

assumes *sum-a-one*: $(\sum i \in S. a \ i) = 1$

assumes *sum-c-one*: $(\sum i \in S. e \ i) = 1$

shows $0 \leq KL\text{-div } S \ a \ e$

unfolding *KL-div-def*

proof –

let $?f = \lambda i. e \ i / a \ i$

have *f-pos*: $\bigwedge i. i \in S \implies 0 < ?f \ i$

using *non-null*

by *simp*

have $a\text{-pos}$: $\bigwedge i. i \in S \implies 0 \leq a i$
using *non-null*
by (*simp add: order.strict-implies-order*)
have $-\log b (\sum_{i \in S}. a i * e i / a i) \leq (\sum_{i \in S}. a i * -\log b (e i / a i))$
using *convex-on-sum[OF fin nemp minus-log-convex[OF b-gt-1]*
 $\text{sum-a-one } a\text{-pos, of } \lambda i. e i / a i]$ *f-pos* **by** *simp*
also have $-\log b (\sum_{i \in S}. a i * e i / a i) = -\log b (\sum_{i \in S}. e i)$
proof –
from *non-null(1)* **have** $\bigwedge i. i \in S \implies a i * e i / a i = e i$ **by** *force*
thus *?thesis* **by** *simp*
qed
finally have $0 \leq (\sum_{i \in S}. a i * -\log b (e i / a i))$
by (*simp add: sum-c-one*)
thus $0 \leq (\sum_{i \in S}. a i * \log b (a i / e i))$
by (*smt (verit, best) b-gt-1 log-divide non-null sum-mono*)
qed

lemma *KL-div-pos-emp*:
 $0 \leq \text{KL-div } \{ \} a e$ **by** (*simp add: KL-div-def*)

lemma *KL-div-pos-gen*:
fixes $a d :: 'b \Rightarrow \text{real}$
assumes *fin: finite S*
assumes *non-null*: $\bigwedge i. i \in S \implies 0 < a i \wedge i. i \in S \implies 0 < d i$
assumes *sum-a-one*: $(\sum_{i \in S}. a i) = 1$
assumes *sum-d-one*: $(\sum_{i \in S}. d i) = 1$
shows $0 \leq \text{KL-div } S a d$
using *KL-div-pos KL-div-pos-emp assms* **by** *metis*

theorem *KL-div-pos2*:
fixes $a d :: 'b \Rightarrow \text{real}$
assumes *fin: finite S*
assumes *non-null*: $\bigwedge i. i \in S \implies 0 \leq a i \wedge i. i \in S \implies 0 < d i$
assumes *sum-a-one*: $(\sum_{i \in S}. a i) = 1$
assumes *sum-c-one*: $(\sum_{i \in S}. d i) = 1$
shows $0 \leq \text{KL-div } S a d$
proof –
have $S = (S \cap \{i. 0 < a i\}) \cup (S \cap \{i. 0 = a i\})$ **using** *non-null(1)* **by**
fastforce
moreover have $(S \cap \{i. 0 < a i\}) \cap (S \cap \{i. 0 = a i\}) = \{ \}$ **by** *auto*
ultimately have
 $\text{eq: } \text{KL-div } S a d = \text{KL-div } (S \cap \{i. 0 < a i\}) a d + \text{KL-div } (S \cap \{i. 0 = a$
 $i\}) a d$
unfolding *KL-div-def*
by (*metis (mono-tags, lifting) fin finite-Un sum.union-disjoint*)
have $\text{KL-div } (S \cap \{i. 0 = a i\}) a d = 0$ **unfolding** *KL-div-def* **by** *simp*
hence $\text{KL-div } S a d = \text{KL-div } (S \cap \{i. 0 < a i\}) a d$ **using** *eq* **by** *simp*
moreover have $0 \leq \text{KL-div } (S \cap \{i. 0 < a i\}) a d$
proof(*cases* $(S \cap \{i. 0 < a i\}) = \{ \}$)

```

case True
thus ?thesis unfolding KL-div-def by simp
next
case False
let ?c = λi. d i / (∑ j ∈ (S ∩ {i. 0 < a i}). d j)
have 1: (∧ i. i ∈ S ∩ {i. 0 < a i} ⇒ 0 < a i) by simp
have 2: (∧ i. i ∈ S ∩ {i. 0 < a i} ⇒ 0 < ?c i)
  by (metis False IntD1 divide-pos-pos fin finite-Int non-null(2) sum-pos)
have 3: (∑ i ∈ (S ∩ {i. 0 < a i}). a i) = 1
  using sum.cong[of S, of S, of (λx. if x ∈ {i. 0 < a i} then a x else 0), of a]
sum.inter-restrict[OF fin, of a] non-null(1) sum-a-one
  by fastforce
have  $(\sum_{i \in S \cap \{j. 0 < a j\}} ?c i) = (\sum_{i \in S \cap \{j. 0 < a j\}} d i) / (\sum_{i \in S \cap \{j. 0 < a j\}} d i)$ 
  by (metis sum-divide-distrib)
hence 5: (∑ i ∈ S ∩ {j. 0 < a j}. ?c i) = 1 using 2 False by force
hence  $0 \leq KL-div (S \cap \{j. 0 < a j\}) a ?c$ 
  using KL-div-pos-gen[
OF finite-Int[OF disjI1, of S, of {j. 0 < a j}], of a, of ?c
] 1 2 3
  by (metis fin)
have fstdb: 0 < (∑ i ∈ S ∩ {i. 0 < a i}. d i) using non-null(2) False
  by (metis Int-Collect fin finite-Int sum-pos)
have 6: 0 ≤ KL-div (S ∩ {i. 0 < a i}) a (λi. d i / (∑ i ∈ (S ∩ {i. 0 < a i}).
d i))
  using 2 3 5
  KL-div-pos-gen[
OF finite-Int[OF disjI1, OF fin], of {i. 0 < a i}, of a, of ?c
]
  by simp
hence
 $KL-div (S \cap \{j. 0 < a j\}) a (\lambda i. d i / (\sum_{i \in (S \cap \{i. 0 < a i\}). d i}) \leq$ 
 $KL-div (S \cap \{j. 0 < a j\}) a d$ 
  using non-null sum.inter-restrict[OF fin, of d, of {i. 0 < a i}]
sum-mono[of S, of (λx. if x ∈ {i. 0 < a i} then d x else 0), of d] non-null(2)
sum-c-one
non-null(2) fstdb KL-div-mul
  by force
moreover have  $0 \leq KL-div (S \cap \{j. 0 < a j\}) a (\lambda i. d i / (\sum_{i \in (S \cap \{i. 0 < a i\}). d i})$ 
 $0 < a i}). d i)$ 
  using KL-div-pos-gen[ OF finite-Int[OF disjI1, OF fin]] using 2 3 5 by
fastforce
ultimately show  $0 \leq KL-div (S \cap \{j. 0 < a j\}) a d$  by simp
qed
ultimately show ?thesis by simp
qed

lemma sum-div-1:
fixes f::'b ⇒ 'c::field

```

assumes $(\sum_{i \in A}. f i) \neq 0$
shows $(\sum_{i \in A}. f i / (\sum_{j \in A}. f j)) = 1$
by (*metis (no-types) assms right-inverse-eq sum-divide-distrib*)

theorem *rate-lower-bound:*

shows $\mathcal{H}(X) \leq \text{code-rate enc } X$

proof –

let $?cr = \text{code-rate enc } X$
let $?r = (\lambda i. 1 / ((b \text{ powr } cw\text{-len } i) * \mathcal{K}))$
have *pos-pi*: $\bigwedge i. i \in L \implies 0 \leq f i$ **using** *fi-pos* **by** *simp*
{
fix i
assume $i \in L$
hence
 $f i * (\log b (1 / (1 / b \text{ powr } (cw\text{-len } i))) + \log b (f i))$
 $= f i * \log b (f i / (1 / b \text{ powr } (cw\text{-len } i)))$
using *log-mult-ext2* [*OF pos-pi, of i*] *b-gt-1*
by *simp (simp add: algebra-simps)*
}
hence *eqpi*:
 $\bigwedge i. i \in L \implies f i * (\log b (1 / (1 / b \text{ powr } (cw\text{-len } i))) + \log b (f i))$
 $= f i * \log b (f i / (1 / b \text{ powr } (cw\text{-len } i)))$
by *simp*
have *sum-one-L*: $(\sum_{i \in L}. f i) = 1$
using *simple-distributed-sum-space*[*OF distr-i*] **by** (*simp add: L-def*)
{
fix i
assume $i \in L$
hence *h1*: $0 \leq f i$ **using** *pos-pi* **by** *blast*
have *h2*: $0 < \mathcal{K} / (1/b \text{ powr } cw\text{-len } i)$ **using** *b-gt-1* *K-pos* **by** *auto*
have *h3*: $0 < 1 / \mathcal{K}$ **using** *K-pos* **by** *simp*
have
 $f i * \log b (f i * \mathcal{K} / (1/b \text{ powr } cw\text{-len } i) * (1 / \mathcal{K})) =$
 $f i * \log b (f i * \mathcal{K} / (1/b \text{ powr } cw\text{-len } i)) + f i * \log b (1 / \mathcal{K})$
using *log-mult-ext3*[*OF h1 h2 h3*]
by (*metis times-divide-eq-right*)
} **hence** *big-eq*:
 $\bigwedge i. i \in L \implies f i * \log b (f i * \mathcal{K} / (1/b \text{ powr } cw\text{-len } i) * (1 / \mathcal{K})) =$
 $f i * \log b (f i * \mathcal{K} / (1/b \text{ powr } cw\text{-len } i)) + f i * \log b (1 / \mathcal{K})$
by (*simp add: inverse-eq-divide*)
have *1*: $?cr - \mathcal{H}(X) = (\sum_{i \in L}. f i * cw\text{-len } i) + (\sum_{i \in L}. f i * \log b (f i))$
using *K-def entropy-rw cr-rw L-def* **by** *simp*
also have *2*: $(\sum_{i \in L}. f i * cw\text{-len } i) = (\sum_{i \in L}. f i * (-\log b (1 / (b \text{ powr } (cw\text{-len } i))))))$
using *b-gt-1 log-divide* **by** *simp*
also have $\dots = -1 * (\sum_{i \in L}. f i * (\log b (1 / (b \text{ powr } (cw\text{-len } i))))))$
using *sum-distrib-left*[*of -1*] $(\lambda i. f i * (-1 * \log b (1 / b \text{ powr } (cw\text{-len } i))))$
L]
by *simp*

finally have
 $?cr - \mathcal{H}(X) = -(\sum i \in L. fi\ i * \log b (1/b\ \text{powr}\ cw\text{-len}\ i)) + (\sum i \in L. fi\ i * \log b (fi\ i))$
by simp
have $?cr - \mathcal{H}(X) = (\sum i \in L. fi\ i * ((\log b (1/ (1/(b\ \text{powr}\ (cw\text{-len}\ i)))))) + \log b (fi\ i))$
using b-gt-1 1
by (simp add: distrib-left sum.distrib)
also have $\dots = (\sum i \in L. fi\ i * ((\log b (fi\ i / (1/(b\ \text{powr}\ (cw\text{-len}\ i))))))$
using Finite-Cartesian-Product.sum-cong-aux[OF eqpi] by simp
also from big-eq have
 $\dots = (\sum i \in L. fi\ i * (\log b (fi\ i * \mathcal{K} / (1 / b\ \text{powr}\ (cw\text{-len}\ i)))) + (\sum i \in L. fi\ i) * \log b (1 / \mathcal{K}))$
using K-pos
by (simp add: sum-distrib-right sum.distrib)
also have $\dots = (\sum i \in L. fi\ i * (\log b (fi\ i * \mathcal{K} / (1 / b\ \text{powr}\ (cw\text{-len}\ i)))) - \log b (\mathcal{K}))$
using K-pos
by (simp add: log-inverse divide-inverse sum-one-L)
also have $\dots = (\sum i \in L. fi\ i * \log b (fi\ i / ?r\ i)) - \log b (\mathcal{K})$
by (metis (mono-tags, opaque-lifting) divide-divide-eq-left divide-divide-eq-right)
also have $\dots = KL\text{-div}\ L\ fi\ ?r - \log b (\mathcal{K})$
using b-gt-1 K-pos log-inverse KL-div-def
by simp
also have $\dots = KL\text{-div}\ L\ fi\ ?r + \log b (1 / \mathcal{K})$
using log-inverse b-val K-pos
by (simp add: inverse-eq-divide)
finally have code-ent-kl-log: ?cr - \mathcal{H}(X) = KL\text{-div}\ L\ fi\ ?r + \log b (1 / \mathcal{K}) **by simp**
have $(\sum i \in L. ?r\ i) = 1$
using sum-div-1[of \lambda i. 1 / (b\ \text{powr}\ (cw\text{-len}\ i))] K-pos K-pow
by simp
moreover have $\bigwedge i. 0 < ?r\ i$ **using b-gt-1 K-pos by simp**
moreover have $(\sum i \in L. fi\ i) = 1$ **using sum-one-L by simp**
ultimately have $0 \leq KL\text{-div}\ L\ fi\ ?r$
using KL-div-pos2[OF fin-L fi-pos] by simp
hence $\log b (1 / \mathcal{K}) \leq ?cr - \mathcal{H}(X)$ **using code-ent-kl-log by simp**
moreover from McMillan have $0 \leq \log b (1 / \mathcal{K})$
using K-pos
by (simp add: b-gt-1)
ultimately show ?thesis by simp
qed
end
end

References

- [1] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.