

Concentrated Liquidity Market Making Operations

Mnacho Echenim

February 10, 2026

Abstract

Automated Market Makers (AMMs) are one of the cornerstones of decentralized finance. They enable users to exchange tokens without the need for order books as would be the case in traditional finance. They involve *liquidity providers*, whose tokens, usually called the *quote* and *base* tokens, can be used in the swap process in exchange for a fee, and *liquidity takers* who swap their tokens. The rules specifying the quantities of tokens that can be swapped and those that act as fees are predefined and lead to several categories of AMMs.

Uniswap v3 introduced a new market-making design that improves capital efficiency by allowing liquidity providers to allocate their assets within selected price intervals. By concentrating liquidity over narrower ranges, providers may earn higher fee income than in earlier AMMs, where liquidity is generally distributed uniformly across all prices. Owing to its success, this design was adopted by several decentralized exchanges on various blockchains, including Trader Joe, PancakeSwap v3, Sunswap v3, and Sushiswap v3. These protocols are collectively referred to as *Concentrated Liquidity Market Makers* (CLMMs). Despite differences in implementation details, such as fee structures, tick spacing, or incentive mechanisms, they all rely on the same underlying principles.

In practice, liquidity takers can thus interact with multiple CLMM pools involving the same pair of tokens but different liquidity profiles or fee structures. A crucial task for them is to understand how these pools can be combined, both conceptually and computationally.

Based on the work in [1], we formalize several notions related to CLMMs, and introduce several operations on such pools that permit to derive an optimality result: if two pools admit the same fees, then the defined transformations permit to determine the optimal quantities of quote tokens to trade in each pool in order to recover as many base tokens as possible.

Contents

1	Preliminary definitions and results	3
1.1	Misc	3
1.2	Support of a discrete function	8

2	Grid information	11
2.1	Definitions	11
2.2	Gross and net token quantities	13
2.2.1	General definitions	13
2.2.2	Finite support restriction	14
2.3	Gross and net quantities of quote tokens	15
2.3.1	Generic functions for quote tokens	15
2.3.2	Finite support restriction	16
2.4	Gross quote token quantity into a pool	17
2.4.1	Function specialization	17
2.4.2	Restriction to pools with a finite liquidity	18
2.5	Net quote token quantity in a pool	21
2.5.1	Function specialization	21
2.5.2	Restriction to pools with a finite liquidity	21
2.6	Gross and net quantities of base tokens	22
2.6.1	Generic functions for base tokens	22
2.6.2	Finite support restriction	24
2.7	Gross base token quantity in a pool	25
2.7.1	Function specialization	25
2.7.2	Restriction to pools with a finite liquidity	26
2.8	Net base token quantity in a pool	27
2.8.1	Function specialization	27
2.8.2	Restriction to pools with a finite liquidity	28
2.9	Swapping tokens, market depth and slippage	28
2.10	Identical profiles	29
3	Grid refinement	31
3.1	Encompassement properties	31
3.2	Finer price grids	32
3.3	Pools with finer grids and coinciding profiles	38
3.4	Spanning grids	41
3.5	Spanning grids and finite liquidity	43
4	CLMM description	44
4.1	Preliminary results	44
4.2	Quote token addition and withdrawal in a CLMM	49
4.3	Base token addition and withdrawal in a CLMM	58
4.4	Market depth and slippage for finer CLMMs	65
4.4.1	Finer pools	65
4.4.2	Finer CLMMs with nonzero liquidity	66
5	Inequalities related to fees	67

6	CLMM transformations	73
6.1	CLMM pool refinement	73
6.2	CLMM pool restriction and slice	81
6.3	CLMM pool join	87
6.4	CLMM pool combination	92
6.5	Optimality result on quote tokens	101
theory <i>CLMM-Misc</i> imports <i>HOL-Analysis.Analysis</i>		

begin

1 Preliminary definitions and results

1.1 Misc

lemma *diff-min-le*:
assumes $(a::real) \leq b$
and $x \leq y$
shows $\min x b - \min x a \leq \min y b - \min y a$
 $\langle proof \rangle$

lemma *sum-ex-strict-pos*:
fixes $f g :: 'i \Rightarrow 'a::ordered-cancel-comm-monoid-add$
assumes *finite* A
and $\forall x \in A. 0 \leq f x$
and $\exists a \in A. 0 < f a$
shows $0 < \text{sum } f A$
 $\langle proof \rangle$

lemma *diff-inv-max-le*:
assumes $0 < a$
and $(a::real) \leq b$
and $x \leq y$
shows $\text{inverse } (\max y a) - \text{inverse } (\max y b) \leq$
 $\text{inverse } (\max x a) - \text{inverse } (\max x b)$
 $\langle proof \rangle$

lemma *int-interval-insert*:
fixes $a::int$
assumes $a \leq b$
shows $\{a..< (b+1)\} = \text{insert } b \{a..< b\}$
 $\langle proof \rangle$

lemma *int-telescoping-sum*:
fixes $f::int \Rightarrow 'a::ab-group-add$

assumes $a \leq b$
shows $(\sum i \in \{a..<b\}. (f\ i - f\ (i+1))) = f\ a - (f\ b)$ $\langle proof \rangle$

lemma *int-telescoping-sum'*:
fixes $f::int \Rightarrow 'a::ab-group-add$
assumes $a \leq b$
shows $(\sum i \in \{a..<b\}. (f\ (i+1) - f\ i)) = f\ b - (f\ a)$
 $\langle proof \rangle$

lemma *int-telescoping-sum-le'*:
fixes $f::int \Rightarrow 'a::ab-group-add$
assumes $a \leq b$
shows $(\sum i \in \{a..b\}. (f\ (i+1) - f\ i)) = f\ (b+1) - (f\ a)$
 $\langle proof \rangle$

lemma *diff-sum-dcomp*:
fixes $f::'a \Rightarrow real$
assumes *finite* A
and $A = B \cup C$
and $B \cap C = \{\}$
shows $x + \text{sum } f\ A - (y + \text{sum } f\ B) = x + \text{sum } f\ C - y$
 $\langle proof \rangle$

lemma *sum-remove-el*:
assumes *finite* A
and $x \in A$
and $B = A - \{x\}$
shows $\text{sum } f\ A = f\ x + \text{sum } f\ B$
 $\langle proof \rangle$

lemma *int-set-bdd-above*:
fixes $X::int\ set$
assumes $X \neq \{\}$
and *bdd-above* X
shows $\text{Sup } X \in X \ \forall x \in X. x \leq \text{Sup } X$
 $\langle proof \rangle$

definition *wedge* **where**
 $\text{wedge } f\ (i::int)\ sqp = (\lambda n. \text{ if } n \leq i \text{ then } f\ n \text{ else } f\ (n-1))(i+1:=sqp)$

lemma *wedge-arg-lt[simp]*:
assumes $n \leq i$
shows $\text{wedge } f\ i\ sqp\ n = f\ n$ $\langle proof \rangle$

lemma *wedge-arg-gt[simp]*:
assumes $i+1 < n$
shows $\text{wedge } f\ i\ sqp\ n = f\ (n-1)$ $\langle proof \rangle$

lemma *wedge-arg-eq[simp]*:

shows $\text{wedge } f \ i \ \text{sqp } (i+1) = \text{sqp } \langle \text{proof} \rangle$

lemma *wedge-strict-mono*:
assumes *strict-mono* f
and $f \ i < \text{sqp}$
and $\text{sqp} < f \ (i+1)$
and $g = \text{wedge } f \ i \ \text{sqp}$
shows *strict-mono* g $\langle \text{proof} \rangle$

lemma *wedge-gt*:
assumes $\forall i. x < f \ i$
and $x < \text{sqp}$
shows $\forall i. x < \text{wedge } f \ j \ \text{sqp } i$
 $\langle \text{proof} \rangle$

lemma *wedge-ge*:
assumes $\forall i. x \leq f \ i$
and $x \leq \text{sqp}$
shows $\forall i. x \leq \text{wedge } f \ j \ \text{sqp } i$
 $\langle \text{proof} \rangle$

lemma *wedge-lt*:
assumes $\forall i. f \ i < x$
and $\text{sqp} < x$
shows $\forall i. \text{wedge } f \ j \ \text{sqp } i < x$
 $\langle \text{proof} \rangle$

lemma *wedge-le*:
assumes $\forall i. f \ i \leq x$
and $\text{sqp} \leq x$
shows $\forall i. \text{wedge } f \ j \ \text{sqp } i \leq x$
 $\langle \text{proof} \rangle$

lemma *wedge-images*:
shows $\forall j. \exists k. f \ j = \text{wedge } f \ i \ \text{sqp } k$
 $\langle \text{proof} \rangle$

lemma *wedge-images'*:
assumes $A = \{j. j \leq i\}$
and $B = \{j. i+1 < j\}$
shows $\text{wedge } f \ i \ \text{sqp } k \in f'A \cup (f'((\lambda i. i-1)'B)) \cup \{\text{sqp}\}$
 $\langle \text{proof} \rangle$

lemma *wedge-as-ubound*:
assumes $\forall (r::\text{real}). \exists (i::\text{int}). r < f \ i$
shows $\forall r. \exists k. r < \text{wedge } f \ i \ \text{sqp } k$ $\langle \text{proof} \rangle$

lemma *wedge-as-lbound*:
assumes $\forall (r::\text{real}). \exists (i::\text{int}). f \ i < r$

shows $\forall r > 0. \exists k. \text{wedge } f \ i \ \text{sqp } k < r \ \langle \text{proof} \rangle$

lemma *wedge-arg-prop*:

shows $\{j. P (\text{wedge } f \ i \ \text{sqp } j)\} \subseteq \{j. j \leq i \wedge P (f \ j)\} \cup$
 $\{j. i+1 < j \wedge P (f \ (j-1))\} \cup \{i+1\}$
 $\langle \text{proof} \rangle$

definition *one-cpl* **where**

one-cpl $\phi i = (\lambda(i::\text{int}). (1::\text{real}) - (\phi i \ i))$

definition *gross-fct* **where**

gross-fct $f \ \phi i = (\lambda i. f \ i \ / \ (\text{one-cpl } \phi i \ i))$

lemma *gross-fct-sgn*:

assumes $\phi i \ i < (1::\text{real})$
shows $((0::\text{real}) \leq f \ i) \longleftrightarrow (0 \leq \text{gross-fct } f \ \phi i \ i) \ \langle \text{proof} \rangle$

lemma *gross-fct-nz-eq*:

assumes $\phi i \ i \neq (1::\text{real})$
shows $f \ i = 0 \longleftrightarrow \text{gross-fct } f \ \phi i \ i = 0 \ \langle \text{proof} \rangle$

lemma *gross-fct-cong*:

assumes $f \ a = f' \ b$
and $\phi i \ a = \phi i' \ b$
shows $\text{gross-fct } f \ \phi i \ a = \text{gross-fct } f' \ \phi i' \ b \ \langle \text{proof} \rangle$

lemma *gross-fct-zero-if*:

assumes $f \ a = 0$
shows $\text{gross-fct } f \ \phi i \ a = 0 \ \langle \text{proof} \rangle$

definition *fee-union* **where**

fee-union $(l1::\text{real}) \ l2 \ f1 \ f2 = (l1 * f1 * (1 - f2) + l2 * f2 * (1 - f1)) /$
 $(l1 * (1 - f2) + l2 * (1 - f1))$

lemma *fee-union-pos*:

assumes $0 \leq l1$
and $0 \leq l2$
and $0 \leq f1$
and $0 \leq f2$
and $f1 < 1$
and $f2 < 1$
shows $0 \leq \text{fee-union } l1 \ l2 \ f1 \ f2 \ \langle \text{proof} \rangle$

lemma *fee-union-lt-1*:

assumes $0 \leq l1$
and $0 \leq l2$
and $0 \leq f1$
and $0 \leq f2$
and $f1 < 1$

and $f2 < 1$
shows $fee\text{-}union\ l1\ l2\ f1\ f2 < 1$
 $\langle proof \rangle$

lemma *diff-inv-le*:
assumes $0 < (x::real)$
and $x \leq y$
and $y \leq z$
shows $(y - x)/(z * z) \leq inverse\ x - inverse\ y$
 $\langle proof \rangle$

lemma *diff-inv-le'*:
assumes $0 < (x::real)$
and $x \leq y$
and $y \leq z$
and $0 \leq a$
shows $a * (y - x)/(z * z) \leq a * (inverse\ x - inverse\ y)$
 $\langle proof \rangle$

lemma *diff-inv-sum-le'*:
assumes $\forall i \in I. (0::real) < f\ i$
and $\forall i \in I. f\ i \leq f\ (i+1)$
and $\forall i \in I. f\ (i+1) \leq z$
and $\forall i \in I. 0 \leq g\ i$
shows $sum\ (\lambda i. g\ i * (f\ (i+1) - f\ i))\ I / (z * z) \leq$
 $sum\ (\lambda i. g\ i * (inverse\ (f\ i) - inverse\ (f\ (i+1))))\ I$
 $\langle proof \rangle$

lemma *diff-inv-ge*:
assumes $0 < (x::real)$
and $x \leq y$
and $y \leq z$
shows $inverse\ y - inverse\ z \leq (z - y)/(x * x)$
 $\langle proof \rangle$

lemma *diff-inv-ge'*:
assumes $0 < (x::real)$
and $x \leq y$
and $y \leq z$
and $0 \leq a$
shows $a * (inverse\ y - inverse\ z) \leq a * (z - y)/(x * x)$
 $\langle proof \rangle$

lemma *diff-inv-sum-ge'*:
assumes $(0::real) < z$
and $\forall i \in I. f\ i \leq f\ (i+1)$
and $\forall i \in I. z \leq f\ i$
and $\forall i \in I. 0 \leq g\ i$
shows $sum\ (\lambda i. g\ i * (inverse\ (f\ i) - inverse\ (f\ (i+1))))\ I \leq$

$\text{sum } (\lambda i. g \ i * (f \ (i+1) - f \ i)) \ I \ / \ (z * z)$
 $\langle \text{proof} \rangle$

1.2 Support of a discrete function

definition *nz-support* where
 $\text{nz-support } f = \{i. f \ i \neq 0\}$

lemma *nz-supportD*:
assumes $i \in \text{nz-support } f$
shows $f \ i \neq 0$ $\langle \text{proof} \rangle$

lemma *wedge-finite-nz-support*:
assumes $\text{finite } (\text{nz-support } f)$
shows $\text{finite } (\text{nz-support } (\text{wedge } f \ i \ \text{sqp}))$
 $\langle \text{proof} \rangle$

lemma *gross-nz-support-eq*:
assumes $\forall i \in \text{nz-support } f. \ \text{phi } i \neq 1$
shows $\text{nz-support } f = \text{nz-support } (\text{gross-fct } f \ \text{phi})$
 $\langle \text{proof} \rangle$

definition *idx-min* where
 $\text{idx-min } f = \text{Inf } (\text{nz-support } f)$

definition *idx-max* where
 $\text{idx-max } f = \text{Sup } (\text{nz-support } f)$

lemma *idx-max-bdd-above-ge*:
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $f \ i \neq 0$
and $\text{bdd-above } (\text{nz-support } f)$
shows $i \leq \text{idx-max } f$
 $\langle \text{proof} \rangle$

lemma *idx-min-bdd-below-le*:
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $f \ i \neq 0$
and $\text{bdd-below } (\text{nz-support } f)$
shows $\text{idx-min } f \leq i$
 $\langle \text{proof} \rangle$

lemma *idx-max-finite-ge*:
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $f \ i \neq 0$
and $\text{finite } (\text{nz-support } f)$
shows $i \leq \text{idx-max } f$ $\langle \text{proof} \rangle$

lemma *idx-min-finite-le*:

fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $f\ i \neq 0$
and $\text{finite } (\text{nz-support } f)$
shows $\text{idx-min } f \leq i$ $\langle \text{proof} \rangle$

lemma idx-max-finite :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{nz-support } f \neq \{\}$
and $\text{finite } (\text{nz-support } f)$
shows $\text{idx-max } f = \text{Max } (\text{nz-support } f)$ $\langle \text{proof} \rangle$

lemma idx-min-finite :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{nz-support } f \neq \{\}$
and $\text{finite } (\text{nz-support } f)$
shows $\text{idx-min } f = \text{Min } (\text{nz-support } f)$ $\langle \text{proof} \rangle$

lemma idx-max-finite-in :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{nz-support } f \neq \{\}$
and $\text{finite } (\text{nz-support } f)$
shows $f\ (\text{idx-max } f) \neq 0$ $\langle \text{proof} \rangle$

lemma idx-min-finite-in :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{nz-support } f \neq \{\}$
and $\text{finite } (\text{nz-support } f)$
shows $f\ (\text{idx-min } f) \neq 0$ $\langle \text{proof} \rangle$

lemma idx-max-finite-gt :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{finite } (\text{nz-support } f)$
and $\text{idx-max } f < i$
shows $f\ i = 0$
 $\langle \text{proof} \rangle$

lemma idx-min-finite-lt :
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{finite } (\text{nz-support } f)$
and $i < \text{idx-min } f$
shows $f\ i = 0$
 $\langle \text{proof} \rangle$

lemma $\text{idx-min-finite-max}$:
fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
assumes $\text{nz-support } f \neq \{\}$
and $\text{finite } (\text{nz-support } f)$
and $\bigwedge j. j < i \implies f\ j = 0$
shows $i \leq \text{idx-min } f$

$\langle \text{proof} \rangle$

lemma *idx-min-max-finite*:
 fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
 assumes $\text{nz-support } f \neq \{\}$
 and $\text{finite } (\text{nz-support } f)$
 shows $\text{idx-min } f \leq \text{idx-max } f$
 $\langle \text{proof} \rangle$

lemma *idx-min-finiteI*:
 fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
 assumes $\text{finite } (\text{nz-support } f)$
 and $f\ i \neq 0$
 and $\bigwedge j. j < i \implies f\ j = 0$
 shows $i = \text{idx-min } f$
 $\langle \text{proof} \rangle$

lemma *idx-min-finite-ge*:
 fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
 assumes $\text{finite } (\text{nz-support } f)$
 and $\text{nz-support } f \neq \{\}$
 and $\bigwedge j. j \leq i \implies f\ j = 0$
 shows $i \leq \text{idx-min } f$
 $\langle \text{proof} \rangle$

lemma *idx-max-finiteI*:
 fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
 assumes $\text{finite } (\text{nz-support } f)$
 and $f\ i \neq 0$
 and $\bigwedge j. j > i \implies f\ j = 0$
 shows $i = \text{idx-max } f$
 $\langle \text{proof} \rangle$

lemma *idx-max-finite-le*:
 fixes $f::'a::\text{conditionally-complete-linorder} \Rightarrow 'b::\text{zero}$
 assumes $\text{finite } (\text{nz-support } f)$
 and $\text{nz-support } f \neq \{\}$
 and $\bigwedge j. i \leq j \implies f\ j = 0$
 shows $\text{idx-max } f \leq i$
 $\langle \text{proof} \rangle$

definition *idx-min-img* **where**
 $\text{idx-min-img } g\ f = g\ (\text{idx-min } f)$

lemma *idx-min-img-eq*:
 assumes $\forall x. f\ x = 0 \longleftrightarrow f'\ x = 0$
 shows $\text{idx-min-img } g\ f = \text{idx-min-img } g\ f' \langle \text{proof} \rangle$

definition *idx-max-img* **where**

$idx-max-imag\ g\ f = g\ (idx-max\ f + 1)$

lemma *idx-max-imag-eq*:
assumes $\forall x. f\ x = 0 \longleftrightarrow f'\ x = 0$
shows $idx-max-imag\ g\ f = idx-max-imag\ g\ f'$ *<proof>*

lemma *non-zero-liq-interv*:
fixes $i::'a::conditionally-complete-linorder$
assumes *finite* (*nz-support* L)
and $L\ i \neq 0$
shows $i \in \{idx-min\ L .. idx-max\ L\}$
<proof>

end
theory *Grid-Information* **imports** *CLMM-Misc*

begin

2 Grid information

2.1 Definitions

A grid information consists of three functions defining the way a grid is associated to (square root) prices, the liquidity on each price range and the fees on each price range.

type-synonym *grid-info* = $(int \Rightarrow real) \times (int \Rightarrow real) \times (int \Rightarrow real)$

definition *grd::grid-info* $\Rightarrow (int \Rightarrow real)$ **where**
grd $P = fst\ P$

definition *lq::grid-info* $\Rightarrow (int \Rightarrow real)$ **where**
lq $P = fst\ (snd\ P)$

definition *fee::grid-info* $\Rightarrow (int \Rightarrow real)$ **where**
fee $P = snd\ (snd\ P)$

Although several results are formalized in a generalized setting, the pools of interest are those admitting a finite range with nonzero liquidity.

definition *finite-liq* **where**
finite-liq $P \longleftrightarrow finite\ (nz-support\ (lq\ P))$

lemma *finite-liqI[intro]*:
assumes $finite\ \{i. lq\ P\ i \neq 0\}$

shows *finite-liq* P $\langle proof \rangle$

lemma *finite-liqD*:
assumes *finite-liq* P
shows *finite* $\{i. \text{liq } P \ i \neq 0\}$ $\langle proof \rangle$

definition *grd-min* **where**
 $grd\text{-}min \ P = idx\text{-}min\text{-}img \ (grd \ P) \ (lq \ P)$

definition *grd-max* **where**
 $grd\text{-}max \ P = idx\text{-}max\text{-}img \ (grd \ P) \ (lq \ P)$

lemma *grd-min-pos*:
assumes *nz-support* $(lq \ P) \neq \{\}$
and $\bigwedge i. 0 < grd \ P \ i$
shows $0 < grd\text{-}min \ P$
 $\langle proof \rangle$

lemma *grd-max-gt*:
assumes *nz-support* $(lq \ P) \neq \{\}$
and $\bigwedge i. 0 < grd \ P \ i$
shows $0 < grd\text{-}max \ P$
 $\langle proof \rangle$

locale *finite-nz-support* =
fixes $L::int \Rightarrow real$
assumes *fin-nz-sup*: *finite* $(nz\text{-}support \ L)$

locale *finite-liq-pool* =
fixes P
assumes *fin-liq*: *finite-liq* P

sublocale *finite-liq-pool* $\subseteq$ *finite-nz-support* $lq \ P$
 $\langle proof \rangle$

context *finite-liq-pool*
begin

lemma *idx-max-mem*:
assumes *nz-support* $(lq \ P) \neq \{\}$
shows *idx-max* $(lq \ P) \in nz\text{-}support \ (lq \ P)$
 $\langle proof \rangle$

lemma *idx-min-mem*:
assumes *nz-support* $(lq \ P) \neq \{\}$
shows *idx-min* $(lq \ P) \in nz\text{-}support \ (lq \ P)$
 $\langle proof \rangle$

lemma *grd-min-max*:

assumes $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{mono } (grd \ P)$
shows $\text{grd-min } P \leq \text{grd-max } P$
 $\langle \text{proof} \rangle$

lemma *finite-liq-gross-fct*:
shows $\text{finite } \{i. \text{gross-fct } (lq \ P) \ (\text{fee } P) \ i \neq 0\}$
 $\langle \text{proof} \rangle$

end

2.2 Gross and net token quantities

2.2.1 General definitions

We define generic functions that are afterwards instantiated to represent the gross (resp. net) quantities of base (resp. quote) tokens in a pool.

definition *rng-token where*
 $\text{rng-token} = (\lambda \text{dff } L \ (pi::\text{real}) \ i. ((L \ i)::\text{real}) * (\text{dff } pi \ (i::\text{int})))$

lemma *rng-token-pos*:
assumes $0 \leq L \ i$
and $0 \leq \text{dff } x \ i$
shows $0 \leq \text{rng-token } \text{dff } L \ x \ i \ \langle \text{proof} \rangle$

lemma *rng-token-continuous-on*:
assumes $\text{continuous-on } A \ (\lambda pi. \text{dff } pi \ i)$
shows $\text{continuous-on } A \ (\lambda pi. \text{rng-token } \text{dff } L \ pi \ i) \ \langle \text{proof} \rangle$

(Anti)-monotonicity is preserved by the generic function *rng-token*.

lemma *rng-token-mono*:
assumes $0 \leq L \ i$
and $\text{mono } (\lambda pi. \text{dff } pi \ i)$
shows $\text{mono } (\lambda pi. \text{rng-token } \text{dff } L \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-token-strict-mono*:
assumes $(0::\text{real}) < L \ i$
and $\text{strict-mono } (\lambda pi. \text{dff } pi \ i)$
shows $\text{strict-mono } (\lambda pi. \text{rng-token } \text{dff } L \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-token-antimono*:
assumes $0 \leq L \ i$
and $\text{antimono } (\lambda pi. \text{dff } pi \ i)$
shows $\text{antimono } (\lambda pi. \text{rng-token } \text{dff } L \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-token-add*:

assumes $\forall i. L\ i = L1\ i + L2\ i$
shows $rng\text{-}token\ dff\ L\ x\ i = rng\text{-}token\ dff\ L1\ x\ i +$
 $rng\text{-}token\ dff\ L2\ x\ i$
 $\langle proof \rangle$

The generic function for the gross or net token quantities on the entire pool is obtained by summation of *rng-token* on all ranges.

definition *gen-token where*
 $gen\text{-}token = (\lambda dff\ L\ pi. (infsum\ (rng\text{-}token\ dff\ L\ pi)\ UNIV))$

lemma *gen-token-pos:*
assumes $\forall i. 0 \leq L\ i$
and $\forall i. 0 \leq dff\ x\ i$
shows $0 \leq gen\text{-}token\ dff\ L\ x\ \langle proof \rangle$

lemma *gen-token-mono:*
assumes $\forall i. 0 \leq L\ i$
and $\forall x. rng\text{-}token\ dff\ L\ x\ summable\text{-}on\ UNIV$
and $\forall i. mono\ (\lambda pi. dff\ pi\ i)$
shows $mono\ (\lambda pi. gen\text{-}token\ dff\ L\ pi)$
 $\langle proof \rangle$

lemma *gen-token-antimono:*
assumes $\forall i. 0 \leq L\ i$
and $\forall x. rng\text{-}token\ dff\ L\ x\ summable\text{-}on\ UNIV$
and $\forall i. antimono\ (\lambda pi. dff\ pi\ i)$
shows $antimono\ (\lambda pi. gen\text{-}token\ dff\ L\ pi)$
 $\langle proof \rangle$

2.2.2 Finite support restriction

context *finite-nz-support*

begin

lemma *finite-nonzero-summable:*
shows $rng\text{-}token\ dff\ L\ x\ summable\text{-}on\ UNIV$
 $\langle proof \rangle$

lemma *gen-token-antimono-finite:*
assumes $\forall i. 0 \leq L\ i$
and $\forall i. antimono\ (\lambda pi. dff\ pi\ i)$
shows $antimono\ (\lambda pi. gen\text{-}token\ dff\ L\ pi)$
 $\langle proof \rangle$

lemma *gen-token-sum:*
shows $gen\text{-}token\ dff\ L\ x =$
 $sum\ (rng\text{-}token\ dff\ L\ x)\ \{i. L\ i \neq 0\}$
 $\langle proof \rangle$

lemma *gen-token-continuous*:
 assumes $\forall i. \text{continuous-on } A \ (\lambda pi. \text{dff } pi \ i)$
 shows *continuous-on* $A \ (\text{gen-token dff } L)$
 $\langle \text{proof} \rangle$

lemma *gen-token-strict-mono*:
 assumes $\forall i. 0 \leq L \ i$
 and *nz-support* $L \neq \{\}$
 and $\forall i. \text{strict-mono} \ (\lambda pi. \text{dff } pi \ i)$
 shows *strict-mono* $(\lambda pi. \text{gen-token dff } L \ pi)$
 $\langle \text{proof} \rangle$

lemma *gen-token-add*:
 assumes $\forall i. L \ i = L1 \ i + L2 \ i$
 and $\forall i. 0 \leq L1 \ i$
 and $\forall i. 0 \leq L2 \ i$
 shows *gen-token dff* $L \ x = \text{gen-token dff } L1 \ x + \text{gen-token dff } L2 \ x$
 $\langle \text{proof} \rangle$

end

2.3 Gross and net quantities of quote tokens

2.3.1 Generic functions for quote tokens

definition *gamma-min-diff* **where**
 $\text{gamma-min-diff } \text{gamma} =$
 $(\lambda(pi::\text{real}) \ i. (\min pi \ (\text{gamma } (i+(1::\text{int})))) - (\min pi \ (\text{gamma } i)))$

lemma *gamma-min-diff-pos*:
 assumes $\text{gamma } i \leq \text{gamma } (i+1)$
 shows $0 \leq \text{gamma-min-diff } \text{gamma } x \ i$
 $\langle \text{proof} \rangle$

lemma *gamma-min-diff-continuous*:
 shows *continuous-on* $A \ (\lambda(pi::\text{real}). \text{gamma-min-diff } \text{gamma } pi \ i)$
 $\langle \text{proof} \rangle$

lemma *gamma-min-diff-mono*:
 fixes $\text{gamma}::\text{int} \Rightarrow \text{real}$
 assumes $\text{gamma } i \leq \text{gamma } (i+1)$
 shows *mono* $(\lambda pi. \text{gamma-min-diff } \text{gamma } pi \ i)$
 $\langle \text{proof} \rangle$

definition *rng-gen-quote* **where**
 $\text{rng-gen-quote } \text{gamma} = (\lambda L \ pi \ i. \text{rng-token } (\text{gamma-min-diff } \text{gamma}) \ L \ pi \ i)$

lemma *rng-gen-quote-pos*:
 assumes $0 \leq L \ i$

and $\text{gamma } i \leq \text{gamma } (i+1)$
shows $0 \leq \text{rng-gen-quote gamma } L \ x \ i \ \langle \text{proof} \rangle$

lemma *rng-gen-quote-continuous-on*:
shows *continuous-on* $A \ (\lambda pi. \text{rng-gen-quote gamma } L \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-gen-quote-mono*:
assumes $0 \leq L \ i$
and $\text{gamma } i \leq \text{gamma } (i+1)$
shows *mono* $(\lambda pi. \text{rng-gen-quote gamma } L \ pi \ i)$
 $\langle \text{proof} \rangle$

definition *gen-quote where*
 $\text{gen-quote} = (\lambda \text{gamma } L \ pi. \text{gen-token } (\text{gamma-min-diff gamma}) \ L \ pi)$

lemma *gen-quote-zero*:
assumes *mono gamma*
and $\bigwedge i. \text{gamma } i < \text{sqr} \implies L \ i = 0$
shows $\text{gen-quote gamma } L \ \text{sqr} = 0 \ \langle \text{proof} \rangle$

lemma *gen-quote-pos*:
assumes $\forall i. 0 \leq L \ i$
and $\forall i. \text{gamma } i \leq \text{gamma } (i+1)$
shows $0 \leq \text{gen-quote gamma } L \ x \ \langle \text{proof} \rangle$

lemma *gen-quote-mono*:
assumes $\forall i. 0 \leq L \ i$
and $\forall x. \text{rng-token } (\text{gamma-min-diff gamma}) \ L \ x \ \text{summable-on UNIV}$
and $\forall i. \text{gamma } i \leq \text{gamma } (i+1)$
shows *mono* $(\text{gen-quote gamma } L)$ $\langle \text{proof} \rangle$

2.3.2 Finite support restriction

context *finite-nz-support*
begin

lemma *gen-quote-mono-finite*:
assumes $\forall i. 0 \leq L \ i$
and $\forall i. \text{gamma } i \leq \text{gamma } (i+1)$
shows *mono* $(\text{gen-quote gamma } L)$
 $\langle \text{proof} \rangle$

lemma *gen-quote-continuous*:
shows *continuous-on* $A \ (\text{gen-quote gamma } L) \ \langle \text{proof} \rangle$

lemma *gen-quote-IVT*:
assumes $(\text{idx-min-img gamma } L) \leq (\text{idx-max-img gamma } L)$
and $\text{gen-quote gamma } L \ (\text{idx-min-img gamma } L) \leq y$

and $y \leq \text{gen-quote } \gamma L (\text{idx-max-img } \gamma L)$
shows $\exists pi \geq (\text{idx-min-img } \gamma L). pi \leq \text{idx-max-img } \gamma L \wedge$
 $\text{gen-quote } \gamma L pi = y$
 $\langle \text{proof} \rangle$

lemma *gen-quote-ne*:

assumes $(\text{idx-min-img } \gamma L) \leq (\text{idx-max-img } \gamma L)$
and $\text{gen-quote } \gamma L (\text{idx-min-img } \gamma L) \leq y$
and $y \leq \text{gen-quote } \gamma L (\text{idx-max-img } \gamma L)$
shows $(\text{gen-quote } \gamma L) - \{y\} \neq \{\}$ $\langle \text{proof} \rangle$

lemma *finite-support-sum*:

assumes $\bigwedge i. L i = 0 \implies f L i = 0$
shows $\text{infsum } (\text{rng-token } \text{dff } (f L) x) \text{ UNIV} =$
 $\text{sum } (\text{rng-token } \text{dff } (f L) x) \{i. L i \neq 0\}$
 $\langle \text{proof} \rangle$

lemma *gen-quote-plus*:

assumes $\forall i. L i = L1 i + L2 i$
and $\forall i. 0 \leq L1 i$
and $\forall i. 0 \leq L2 i$
shows $\text{gen-quote } \gamma L x = \text{gen-quote } \gamma L1 x + \text{gen-quote } \gamma L2 x$
 $\langle \text{proof} \rangle$

end

2.4 Gross quote token quantity into a pool

2.4.1 Function specialization

When the quote tokens are added to a pool, fees are to be taken into account: if a user adds a quantity q of tokens into a pool, the computation of the amount of base tokens received is based in $q \cdot (1 - \varphi)$.

definition *rng-quote-gross* **where**

$\text{rng-quote-gross } P = \text{rng-gen-quote } (\text{grd } P) (\text{gross-fct } (lq P) (\text{fee } P))$

lemma *rng-quote-gross-pos*:

assumes $0 \leq \text{gross-fct } (lq P) (\text{fee } P) i$
and $\text{grd } P i \leq \text{grd } P (i+1)$
shows $0 \leq \text{rng-quote-gross } P pi i$ $\langle \text{proof} \rangle$

lemma *rng-quote-gross-continuous-on*:

shows $\text{continuous-on } A (\lambda pi. \text{rng-quote-gross } P pi i)$
 $\langle \text{proof} \rangle$

lemma *rng-quote-gross-mono*:

assumes $0 \leq \text{gross-fct } (lq P) (\text{fee } P) i$
and $\text{grd } P i \leq \text{grd } P (i+1)$
shows $\text{mono } (\lambda pi. \text{rng-quote-gross } P pi i)$ $\langle \text{proof} \rangle$

definition *quote-gross* **where**
quote-gross $P = \text{gen-quote } (\text{grd } P) (\text{gross-fct } (\text{lq } P) (\text{fee } P))$

lemma *quote-gross-pos*:
assumes $\forall i. 0 \leq \text{gross-fct } (\text{lq } P) (\text{fee } P) i$
and $\forall i. \text{grd } P i \leq \text{grd } P (i+1)$
shows $0 \leq \text{quote-gross } P x$ $\langle \text{proof} \rangle$

lemma *quote-gross-mono*:
assumes $\forall i. 0 \leq (\text{lq } P) i$
and $\forall i. (\text{fee } P) i < 1$
and $\forall i. (\text{grd } P) i \leq (\text{grd } P) (i+1)$
and $\forall x. \text{rng-token } (\text{gamma-min-diff } (\text{grd } P)) (\text{gross-fct } (\text{lq } P) (\text{fee } P)) x$
 summable-on UNIV
shows $\text{mono } (\text{quote-gross } P)$ $\langle \text{proof} \rangle$

lemma *gen-quote-grd-min*:
assumes $\text{mono } (\text{grd } P)$
and $\text{finite } (\text{nz-support } L)$
and $\text{nz-support } L \neq \{\}$
and $\text{nz-support } L = \text{nz-support } (\text{lq } P)$
shows $\text{gen-quote } (\text{grd } P) L (\text{grd-min } P) = 0$
 $\langle \text{proof} \rangle$

Definition of the grid point that is reached in a pool for a given gross quantity of quote tokens.

definition *quote-reach* **where**
quote-reach $= (\lambda P y.$
 $\text{if } y = 0 \text{ then } (\text{grd-min } P)$
 $\text{else } \text{Inf } ((\text{quote-gross } P) - \{y\}))$

2.4.2 Restriction to pools with a finite liquidity

context *finite-liq-pool*

begin

lemma *quote-gross-mono-finite*:
assumes $\forall i. 0 \leq (\text{lq } P) i$
and $\forall i. (\text{fee } P) i < 1$
and $\forall i. (\text{grd } P) i \leq (\text{grd } P) (i+1)$
shows $\text{mono } (\text{quote-gross } P)$
 $\langle \text{proof} \rangle$

lemma *quote-gross-mono-finite'*:
assumes $\forall i. 0 \leq (\text{lq } P) i$
and $\forall i. (\text{fee } P) i < 1$
and $\text{mono } (\text{grd } P)$

shows *mono* (*quote-gross* *P*)
 ⟨*proof*⟩

lemma *quote-gross-continuous*:
shows *continuous-on* *A* (*quote-gross* *P*) ⟨*proof*⟩

lemma *quote-gross-IVT*:
assumes $\forall i. \text{fee } P \ i \neq 1$
and $\text{grd-min } P \leq \text{grd-max } P$
and $\text{quote-gross } P \ (\text{grd-min } P) \leq y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
shows $\exists pi \geq (\text{grd-min } P). \ pi \leq (\text{grd-max } P) \wedge$
 $\text{quote-gross } P \ pi = y$
 ⟨*proof*⟩

lemma *quote-gross-ne*:
assumes $\forall i. \text{fee } P \ i \neq 1$
and $\text{grd-min } P \leq \text{grd-max } P$
and $\text{quote-gross } P \ (\text{grd-min } P) \leq y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
shows $\text{quote-gross } P - \{y\} \neq \{\}$ ⟨*proof*⟩

lemma *quote-gross-grd-min*:
assumes *mono* (*grd* *P*)
shows $\text{quote-gross } P \ (\text{grd-min } P) = 0$
 ⟨*proof*⟩

lemma *quote-reach-mem*:
assumes $\forall i. 0 \leq \text{lq } P \ i$
and $\forall i. \text{fee } P \ i < 1$
and *mono* (*grd* *P*)
and $0 \leq y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
shows $\text{quote-reach } P \ y \in \text{quote-gross } P - \{y\}$
 ⟨*proof*⟩

lemma *quote-gross-inv-strict-mono*:
assumes *mono* (*quote-gross* *P*)
and $\text{quote-gross } P \ \text{spp}' < y$
and $\text{spp} \in \text{quote-gross } P - \{y\}$
shows $\text{spp}' < \text{spp}$
 ⟨*proof*⟩

lemma *quote-gross-inv-bounded*:
assumes *mono* (*quote-gross* *P*)
and $\text{quote-gross } P \ (\text{grd-min } P) < y$
and $y < \text{quote-gross } P \ (\text{grd-max } P)$
shows $\forall \text{spp}' \in \text{quote-gross } P - \{y\}. \ \text{dist } (\text{grd-min } P) \ \text{spp}' \leq \text{grd-max } P - \text{grd-min } P$

$\langle proof \rangle$

lemma *quote-gross-bdd-below*:

assumes *mono* (*quote-gross* *P*)

and *quote-gross* *P* (*grd-min* *P*) $< y$

shows *bdd-below* (*quote-gross* *P* - $\{y\}$) $\langle proof \rangle$

lemma *quote-reach-le*:

assumes $\forall i. 0 \leq lq\ P\ i$

and $\forall i. fee\ P\ i < 1$

and *mono* (*grd* *P*)

and $0 < y$

and *sqp* $\in quote-gross\ P - \{y\}$

shows *quote-reach* *P* *y* $\leq sqp$

$\langle proof \rangle$

lemma *quote-reach-gross-le*:

assumes $\forall i. 0 \leq lq\ P\ i$

and $\forall i. fee\ P\ i < 1$

and *mono* (*grd* *P*)

and *grd-min* *P* $\leq sqp$

shows *quote-reach* *P* (*quote-gross* *P* *sqp*) $\leq sqp$

$\langle proof \rangle$

lemma *quote-gross-reach-eq*:

assumes $\forall i. 0 \leq lq\ P\ i$

and $\forall i. fee\ P\ i < 1$

and *mono* (*grd* *P*)

and $0 \leq y$

and $y \leq quote-gross\ P\ (grd-max\ P)$

shows *quote-gross* *P* (*quote-reach* *P* *y*) = *y*

$\langle proof \rangle$

lemma *quote-reach-ge*:

assumes $\forall i. 0 \leq lq\ P\ i$

and $\forall i. fee\ P\ i < 1$

and *mono* (*grd* *P*)

and *grd-min* *P* $\leq grd-max\ P$

and $0 < y$

and $y \leq quote-gross\ P\ (grd-max\ P)$

shows *grd-min* *P* $\leq quote-reach\ P\ y$

$\langle proof \rangle$

end

2.5 Net quote token quantity in a pool

2.5.1 Function specialization

There are no fees to take into account when tokens are withdrawn from a pool.

definition *rng-quote-net* **where**

rng-quote-net $P = \text{rng-gen-quote } (\text{grd } P) (lq \ P)$

lemma *rng-quote-net-pos*:

assumes $0 \leq (lq \ P) \ i$

and $\text{grd } P \ i \leq \text{grd } P \ (i+1)$

shows $0 \leq \text{rng-quote-net } P \ x \ i$ $\langle \text{proof} \rangle$

lemma *rng-quote-net-continuous-on*:

shows *continuous-on* $A \ (\lambda pi. \text{rng-quote-net } P \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-quote-net-mono*:

assumes $0 \leq (lq \ P) \ i$

and $\text{grd } P \ i \leq \text{grd } P \ (i+1)$

shows *mono* $(\lambda pi. \text{rng-quote-net } P \ pi \ i)$ $\langle \text{proof} \rangle$

definition *quote-net* **where**

quote-net $P = \text{gen-quote } (\text{grd } P) (lq \ P)$

lemma *quote-net-pos*:

assumes $\forall i. 0 \leq (lq \ P) \ i$

and $\forall i. \text{grd } P \ i \leq \text{grd } P \ (i+1)$

shows $0 \leq \text{quote-net } P \ x$ $\langle \text{proof} \rangle$

lemma *quote-net-mono*:

assumes $\forall i. 0 \leq (lq \ P) \ i$

and $\forall i. (\text{fee } P) \ i < 1$

and $\forall i. (\text{grd } P) \ i \leq (\text{grd } P) \ (i+1)$

and $\forall x. \text{rng-token } (\text{gamma-min-diff } (\text{grd } P)) (lq \ P) \ x \text{ summable-on } UNIV$

shows *mono* $(\text{quote-net } P)$ $\langle \text{proof} \rangle$

2.5.2 Restriction to pools with a finite liquidity

context *finite-liq-pool*

begin

lemma *quote-net-continuous*:

shows *continuous-on* $A \ (\text{quote-net } P)$ $\langle \text{proof} \rangle$

lemma *quote-net-IVT*:

assumes $\forall i. \text{fee } P \ i \neq 1$

and $\text{grd-min } P \leq \text{grd-max } P$

and $\text{quote-net } P \text{ (grd-min } P) \leq y$
and $y \leq \text{quote-net } P \text{ (grd-max } P)$
shows $\exists pi \geq (\text{grd-min } P). pi \leq (\text{grd-max } P) \wedge$
 $\text{quote-net } P pi = y$
 $\langle \text{proof} \rangle$

lemma *quote-net-ne*:
assumes $\forall i. \text{fee } P i \neq 1$
and $\text{grd-min } P \leq \text{grd-max } P$
and $\text{quote-net } P \text{ (grd-min } P) \leq y$
and $y \leq \text{quote-net } P \text{ (grd-max } P)$
shows $\text{quote-net } P - \{y\} \neq \{\}$ $\langle \text{proof} \rangle$

lemma *quote-net-mono-finite-liq*:
assumes $\forall i. 0 \leq (\text{liq } P) i$
and $\forall i. (\text{fee } P) i < 1$
and $\forall i. (\text{grd } P) i \leq (\text{grd } P) (i+1)$
shows *mono* $(\text{quote-net } P)$ $\langle \text{proof} \rangle$

end

2.6 Gross and net quantities of base tokens

2.6.1 Generic functions for base tokens

definition *inv-gamma-max-diff* **where**
 $\text{inv-gamma-max-diff} = (\lambda \text{gamma } (pi::\text{real}) i. \text{inverse } (\max pi (\text{gamma } i)) -$
 $\text{inverse } (\max pi (\text{gamma } (i+(1::\text{int}))))))$

lemma *inv-max-pos*:
assumes $0 < a$
and $(a::\text{real}) \leq b$
shows $0 \leq \text{inverse } (\max x a) - \text{inverse } (\max x b)$
 $\langle \text{proof} \rangle$

lemma *inv-gamma-max-diff-pos*:
assumes $\text{gamma } i \leq \text{gamma } (i+(1::\text{int}))$
and $0 < \text{gamma } i$
shows $0 \leq \text{inv-gamma-max-diff } \text{gamma } x i$ $\langle \text{proof} \rangle$

lemma *inv-gamma-max-diff-continuous*:
assumes $\text{gamma } i \leq \text{gamma } (i+(1::\text{int}))$
and $0 < \text{gamma } i$
shows *continuous-on* $A (\lambda pi. \text{inv-gamma-max-diff } \text{gamma } pi i)$
 $\langle \text{proof} \rangle$

lemma *inv-gamma-max-diff-antimono*:
assumes $\text{gamma } i \leq \text{gamma } (i+(1::\text{int}))$
and $0 < \text{gamma } i$
shows *antimono* $(\lambda pi. \text{inv-gamma-max-diff } \text{gamma } pi i)$

$\langle \text{proof} \rangle$

definition *rng-gen-base* **where**

rng-gen-base =
($\lambda \text{gamma } L \text{ pi } i. \text{rng-token } (\text{inv-gamma-max-diff } \text{gamma}) \text{ } L \text{ pi } i$)

lemma *rng-gen-base-pos*:

assumes $\text{gamma } i \leq \text{gamma } (i + (1::\text{int}))$
and $0 < \text{gamma } i$
and $0 \leq L \text{ } i$
shows $0 \leq \text{rng-gen-base } \text{gamma } L \text{ } x \text{ } i$ $\langle \text{proof} \rangle$

lemma *rng-gen-base-continuous-on*:

assumes $\text{gamma } i \leq \text{gamma } (i + (1::\text{int}))$
and $0 < \text{gamma } i$
shows *continuous-on* $A \text{ } (\lambda \text{pi}. \text{rng-gen-base } \text{gamma } L \text{ pi } i)$ $\langle \text{proof} \rangle$

lemma *rng-gen-base-antimono*:

assumes $\text{gamma } i \leq \text{gamma } (i + (1::\text{int}))$
and $0 < \text{gamma } i$
and $0 \leq L \text{ } i$
shows *antimono* $(\lambda \text{pi}. \text{rng-gen-base } \text{gamma } L \text{ pi } i)$
 $\langle \text{proof} \rangle$

definition *gen-base* **where**

gen-base = ($\lambda \text{gamma } L \text{ pi}. \text{gen-token } (\text{inv-gamma-max-diff } \text{gamma}) \text{ } L \text{ pi}$)

lemma *gen-base-pos*:

assumes $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::\text{int}))$
and $\forall i. 0 < \text{gamma } i$
and $\forall i. 0 \leq L \text{ } i$
shows $0 \leq \text{gen-base } \text{gamma } L \text{ } x$ $\langle \text{proof} \rangle$

lemma *gen-base-antimono*:

assumes $\forall x. \text{rng-token } (\text{inv-gamma-max-diff } \text{gamma}) \text{ } L \text{ } x \text{ summable-on } \text{UNIV}$
and $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::\text{int}))$
and $\forall i. 0 < \text{gamma } i$
and $\forall i. 0 \leq L \text{ } i$
shows *antimono* $(\text{gen-base } \text{gamma } L)$ $\langle \text{proof} \rangle$

lemma *gen-base-zero*:

assumes *mono* gamma
and $\bigwedge i. \text{sqr } < \text{gamma } (i+1) \implies L \text{ } i = 0$
shows *gen-base* $\text{gamma } L \text{ } \text{sqr} = 0$ $\langle \text{proof} \rangle$

lemma *gen-base-grd-max*:

assumes *mono* $(\text{grd } P)$
and *finite* $(\text{nz-support } L)$
and $\text{nz-support } L \neq \{\}$

and $\text{nz-support } L = \text{nz-support } (lq \ P)$
shows $\text{gen-base } (grd \ P) \ L \ (grd\text{-max } P) = 0$
 $\langle \text{proof} \rangle$

2.6.2 Finite support restriction

context *finite-nz-support*
begin

lemma *gen-base-continuous*:

assumes $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::int))$
and $\forall i. 0 < \text{gamma } i$
shows *continuous-on* $A \ (\text{gen-base } \text{gamma } L) \ \langle \text{proof} \rangle$

lemma *gen-base-IVT*:

assumes $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::int))$
and $\forall i. 0 < \text{gamma } i$
and $(\text{idx-min-img } \text{gamma } L) \leq (\text{idx-max-img } \text{gamma } L)$
and $\text{gen-base } \text{gamma } L \ (\text{idx-max-img } \text{gamma } L) \leq y$
and $y \leq \text{gen-base } \text{gamma } L \ (\text{idx-min-img } \text{gamma } L)$
shows $\exists pi \geq (\text{idx-min-img } \text{gamma } L). \ pi \leq (\text{idx-max-img } \text{gamma } L) \wedge$
 $\text{gen-base } \text{gamma } L \ pi = y$
 $\langle \text{proof} \rangle$

lemma *gen-base-ne*:

assumes $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::int))$
and $\forall i. 0 < \text{gamma } i$
and $(\text{idx-min-img } \text{gamma } L) \leq (\text{idx-max-img } \text{gamma } L)$
and $\text{gen-base } \text{gamma } L \ (\text{idx-max-img } \text{gamma } L) \leq y$
and $y \leq \text{gen-base } \text{gamma } L \ (\text{idx-min-img } \text{gamma } L)$
shows $(\text{gen-base } \text{gamma } L) - \{y\} \neq \{\}$ $\langle \text{proof} \rangle$

lemma *gen-base-antimono-finite*:

assumes $\forall i. \text{gamma } i \leq \text{gamma } (i + (1::int))$
and $\forall i. 0 < \text{gamma } i$
and $\forall i. 0 \leq L \ i$
shows *antimono* $(\text{gen-base } \text{gamma } L)$
 $\langle \text{proof} \rangle$

lemma *gen-base-gross*:

assumes $\forall i. L \ i = L1 \ i + L2 \ i$
and $\forall i. 0 \leq L1 \ i$
and $\forall i. 0 \leq L2 \ i$
shows $\text{gen-base } \text{gam } L \ x = \text{gen-base } \text{gam } L1 \ x + \text{gen-base } \text{gam } L2 \ x$
 $\langle \text{proof} \rangle$

end

2.7 Gross base token quantity in a pool

2.7.1 Function specialization

definition *rng-base-gross* **where**

$\text{rng-base-gross } P = \text{rng-gen-base } (\text{grd } P) (\text{gross-fct } (\text{lq } P) (\text{fee } P))$

lemma *rng-base-gross-pos*:

assumes $0 \leq \text{gross-fct } (\text{lq } P) (\text{fee } P) \ i$

and $\text{grd } P \ i \leq \text{grd } P \ (i+1)$

and $0 < \text{grd } P \ i$

shows $0 \leq \text{rng-base-gross } P \ x \ i \ \langle \text{proof} \rangle$

lemma *rng-base-gross-continuous-on*:

assumes $\text{grd } P \ i \leq \text{grd } P \ (i+1)$

and $0 < \text{grd } P \ i$

shows $\text{continuous-on } A \ (\lambda pi. \text{rng-base-gross } P \ pi \ i)$

$\langle \text{proof} \rangle$

lemma *rng-base-gross-mono*:

assumes $0 \leq \text{gross-fct } (\text{lq } P) (\text{fee } P) \ i$

and $\text{grd } P \ i \leq \text{grd } P \ (i+1)$

and $0 < \text{grd } P \ i$

shows $\text{antimono } (\lambda pi. \text{rng-base-gross } P \ pi \ i) \ \langle \text{proof} \rangle$

definition *base-gross* **where**

$\text{base-gross } P = \text{gen-base } (\text{grd } P) (\text{gross-fct } (\text{lq } P) (\text{fee } P))$

lemma *base-gross-pos*:

assumes $\forall i. 0 \leq \text{gross-fct } (\text{lq } P) (\text{fee } P) \ i$

and $\forall i. \text{grd } P \ i \leq \text{grd } P \ (i+1)$

and $\forall i. 0 < \text{grd } P \ i$

shows $0 \leq \text{base-gross } P \ x \ \langle \text{proof} \rangle$

lemma *base-gross-antimono*:

assumes $\forall i. 0 \leq (\text{lq } P) \ i$

and $\forall i. (\text{fee } P) \ i < 1$

and $\forall i. (\text{grd } P) \ i \leq (\text{grd } P) \ (i+1)$

and $\forall i. 0 < \text{grd } P \ i$

and $\forall x. \text{rng-token } (\text{inv-gamma-max-diff } (\text{grd } P)) (\text{gross-fct } (\text{lq } P) (\text{fee } P)) \ x$
 summable-on UNIV

shows $\text{antimono } (\text{base-gross } P) \ \langle \text{proof} \rangle$

lemma *base-gross-grd-max*:

assumes $\text{mono } (\text{grd } P)$

and $\text{finite } (\text{nz-support } (\text{lq } P))$

shows $\text{base-gross } P \ (\text{grd-max } P) = 0$
 $\langle \text{proof} \rangle$

definition *base-reach* **where**

$base\text{-}reach = (\lambda P y.$
 $\quad if\ y = 0$
 $\quad then\ (grd\text{-}max\ P)$
 $\quad else\ Sup\ ((base\text{-}gross\ P) - \{y\}))$

2.7.2 Restriction to pools with a finite liquidity

context *finite-liq-pool*
begin

lemma *base-gross-continuous*:
 $\text{assumes } \forall i. \text{grd } P\ i \leq \text{grd } P\ (i+1)$
 $\text{and } \forall i. 0 < \text{grd } P\ i$
shows *continuous-on* $A\ (base\text{-}gross\ P)\ \langle proof \rangle$

lemma *base-gross-IVT*:
 $\text{assumes } \forall i. \text{grd } P\ i \leq \text{grd } P\ (i+1)$
 $\text{and } \forall i. 0 < \text{grd } P\ i$
 $\text{and } \forall i. fee\ P\ i \neq 1$
 $\text{and } \text{grd}\text{-}min\ P \leq \text{grd}\text{-}max\ P$
 $\text{and } base\text{-}gross\ P\ (\text{grd}\text{-}max\ P) \leq y$
 $\text{and } y \leq base\text{-}gross\ P\ (\text{grd}\text{-}min\ P)$
shows $\exists pi \geq (\text{grd}\text{-}min\ P). \ pi \leq (\text{grd}\text{-}max\ P) \wedge$
 $\quad base\text{-}gross\ P\ pi = y$
 $\langle proof \rangle$

lemma *base-gross-ne*:
 $\text{assumes } \forall i. \text{grd } P\ i \leq \text{grd } P\ (i+1)$
 $\text{and } \forall i. 0 < \text{grd } P\ i$
 $\text{and } \forall i. fee\ P\ i \neq 1$
 $\text{and } \text{grd}\text{-}min\ P \leq \text{grd}\text{-}max\ P$
 $\text{and } base\text{-}gross\ P\ (\text{grd}\text{-}max\ P) \leq y$
 $\text{and } y \leq base\text{-}gross\ P\ (\text{grd}\text{-}min\ P)$
shows $base\text{-}gross\ P - \{y\} \neq \{\}$ $\langle proof \rangle$

lemma *base-gross-antimono-finite*:
 $\text{assumes } \forall i. 0 \leq (lq\ P)\ i$
 $\text{and } \forall i. \text{grd } P\ i \leq \text{grd } P\ (i+1)$
 $\text{and } \forall i. 0 < \text{grd } P\ i$
 $\text{and } \forall i. (fee\ P)\ i < 1$
shows *antimono* $(base\text{-}gross\ P)\ \langle proof \rangle$

lemma *base-reach-mem*:
 $\text{assumes } \forall i. \text{grd } P\ i \leq \text{grd } P\ (i+1)$
 $\text{and } \forall i. 0 < \text{grd } P\ i$
 $\text{and } \forall i. fee\ P\ i < 1$
 $\text{and } \forall i. 0 \leq lq\ P\ i$
 $\text{and } mono\ (\text{grd } P)$
 $\text{and } \text{grd}\text{-}min\ P \leq \text{grd}\text{-}max\ P$

and $0 \leq y$
and $y \leq \text{base-gross } P \text{ (grd-min } P)$
shows $\text{base-reach } P \ y \in \text{base-gross } P - \{y\}$
 $\langle \text{proof} \rangle$

lemma *base-gross-dwn*:
assumes $\forall i. \text{grd } P \ i \leq \text{grd } P \ (i+1)$
and $\forall i. 0 < \text{grd } P \ i$
and $\forall i. \text{fee } P \ i < 1$
and $\forall i. 0 \leq \text{lq } P \ i$
and $\text{mono } (\text{grd } P)$
and $\text{grd-min } P \leq \text{grd-max } P$
and $0 \leq y$
and $y \leq \text{base-gross } P \text{ (grd-min } P)$
shows $\text{base-gross } P \text{ (base-reach } P \ y) = y$
 $\langle \text{proof} \rangle$

end

2.8 Net base token quantity in a pool

2.8.1 Function specialization

definition *rng-base-net* **where**
 $\text{rng-base-net } P = \text{rng-gen-base } (\text{grd } P) \ (\text{lq } P)$

lemma *rng-base-net-pos*:
assumes $\text{grd } P \ i \leq \text{grd } P \ (i + (1::\text{int}))$
and $0 < \text{grd } P \ i$
and $0 \leq \text{lq } P \ i$
shows $0 \leq \text{rng-base-net } P \ x \ i \ \langle \text{proof} \rangle$

lemma *rng-base-net-continuous-on*:
assumes $\text{grd } P \ i \leq \text{grd } P \ (i + (1::\text{int}))$
and $0 < \text{grd } P \ i$
shows $\text{continuous-on } A \ (\lambda pi. \text{rng-base-net } P \ pi \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-base-net-mono*:
assumes $\text{grd } P \ i \leq \text{grd } P \ (i + (1::\text{int}))$
and $0 < \text{grd } P \ i$
and $0 \leq \text{lq } P \ i$
shows $\text{antimono } (\lambda pi. \text{rng-base-net } P \ pi \ i) \ \langle \text{proof} \rangle$

definition *base-net* **where**
 $\text{base-net } P = \text{gen-base } (\text{grd } P) \ (\text{lq } P)$

lemma *base-net-pos*:
assumes $\forall i. \text{grd } P \ i \leq \text{grd } P \ (i + (1::\text{int}))$
and $\forall i. 0 < \text{grd } P \ i$

and $\forall i. 0 \leq lq\ P\ i$
 shows $0 \leq base-net\ P\ x$ $\langle proof \rangle$

2.8.2 Restriction to pools with a finite liquidity

context *finite-liq-pool*
 begin

lemma *base-net-continuous*:
 assumes $\forall i. grd\ P\ i \leq grd\ P\ (i + (1::int))$
 and $\forall i. 0 < grd\ P\ i$
 shows *continuous-on* $A\ (base-net\ P)$ $\langle proof \rangle$

lemma *base-net-IVT*:
 assumes $\forall i. grd\ P\ i \leq grd\ P\ (i + (1::int))$
 and $\forall i. 0 < grd\ P\ i$
 and $grd-min\ P \leq grd-max\ P$
 and $base-net\ P\ (grd-max\ P) \leq y$
 and $y \leq base-net\ P\ (grd-min\ P)$
 shows $\exists pi \geq (grd-min\ P). pi \leq (grd-max\ P) \wedge$
 $base-net\ P\ pi = y$
 $\langle proof \rangle$

lemma *base-net-ne*:
 assumes $\forall i. grd\ P\ i \leq grd\ P\ (i + (1::int))$
 and $\forall i. 0 < grd\ P\ i$
 and $grd-min\ P \leq grd-max\ P$
 and $base-net\ P\ (grd-max\ P) \leq y$
 and $y \leq base-net\ P\ (grd-min\ P)$
 shows $base-net\ P - \{y\} \neq \{\}$ $\langle proof \rangle$

lemma *base-net-antimono-finite*:
 assumes $\forall i. grd\ P\ i \leq grd\ P\ (i + (1::int))$
 and $\forall i. 0 < grd\ P\ i$
 and $\forall i. 0 \leq lq\ P\ i$
 shows *antimono* $(base-net\ P)$ $\langle proof \rangle$

end

2.9 Swapping tokens, market depth and slippage

Given a grid point π and a quantity y of quote tokens to add to the pool, this function computes the amount of base tokens that are retrieved from the pool.

definition *quote-swap* where
 $quote-swap\ P = (\lambda pi\ y.$
 $base-net\ P\ pi - base-net\ P\ (quote-reach\ P\ (y + quote-gross\ P\ pi)))$

Given a grid point π and a quantity x of base tokens to add to the pool,

this function computes the amount of quote tokens that are retrieved from the pool.

definition *base-swap* **where**

base-swap $P = (\lambda pi\ x.$

$\text{quote-net } P\ pi - \text{quote-net } P\ (\text{base-reach } P\ (x + \text{base-gross } P\ pi)))$

The market depth in a pool takes as arguments two grid points π and π' , and returns the amounts of base or quote tokens that have to be added to the pool for its state to get from π to π' .

definition *mkt-depth* **where**

mkt-depth $P = (\lambda\ pi\ pi'. \text{ if } pi < pi' \text{ then } (\text{base-net } P\ pi - \text{base-net } P\ pi') \\ \text{ else } (\text{quote-net } P\ pi - \text{quote-net } P\ pi'))$

Base and quote slippages relate the amount of tokens withdrawn from the pool from those given by an infinitesimally small amount of tokens and that can be deduced from the grid point.

definition *quote-slippage* **where**

quote-slippage $P = (\lambda pi\ y. y / (\text{quote-swap } P\ pi\ y * pi * pi) - 1)$

definition *base-slippage* **where**

base-slippage $P = (\lambda pi\ x. \text{base-swap } P\ pi\ x / (x * pi * pi) - 1)$

2.10 Identical profiles

definition *id-grid-on* **where**

id-grid-on $P\ P'\ I \longleftrightarrow (\forall i \in I. \text{grd } P\ i = \text{grd } P'\ i)$

lemma *id-grid-onI[intro]*:

assumes $\bigwedge i. i \in I \implies \text{grd } P\ i = \text{grd } P'\ i$

shows *id-grid-on* $P\ P'\ I$ $\langle \text{proof} \rangle$

lemma *id-grid-onD[dest]*:

assumes *id-grid-on* $P\ P'\ I$

and $i \in I$

shows $\text{grd } P\ i = \text{grd } P'\ i$ $\langle \text{proof} \rangle$

lemma *id-grid-on-comm*:

assumes *id-grid-on* $P\ P'\ I$

shows *id-grid-on* $P'\ P\ I$

$\langle \text{proof} \rangle$

lemma *id-grid-on-mono*:

assumes *id-grid-on* $P\ P'\ I$

and $I' \subseteq I$

shows *id-grid-on* $P\ P'\ I'$ $\langle \text{proof} \rangle$

definition *same-nz-liq-on* **where**

same-nz-liq-on $P\ P'\ I \longleftrightarrow \text{id-grid-on } P\ P'\ I \wedge$

$$(\forall i \in I. (lq\ P\ i = 0) \longleftrightarrow (lq\ P'\ i = 0))$$

lemma *same-nz-liq-onI*[*intro*]:
assumes *id-grid-on* $P\ P'\ I$
and $\bigwedge i. i \in I \implies ((lq\ P\ i = 0) \longleftrightarrow (lq\ P'\ i = 0))$
shows *same-nz-liq-on* $P\ P'\ I$ $\langle proof \rangle$

lemma *same-nz-liq-onD*[*dest*]:
assumes *same-nz-liq-on* $P\ P'\ I$
and $i \in I$
shows $grd\ P\ i = grd\ P'\ i \ (lq\ P\ i = 0) \longleftrightarrow (lq\ P'\ i = 0)$
 $\langle proof \rangle$

lemma *same-nz-liq-on-comm*:
assumes *same-nz-liq-on* $P\ P'\ I$
shows *same-nz-liq-on* $P'\ P\ I$
 $\langle proof \rangle$

lemma *same-nz-liq-on-mono*:
assumes *same-nz-liq-on* $P\ P'\ I$
and $I' \subseteq I$
shows *same-nz-liq-on* $P\ P'\ I'$
 $\langle proof \rangle$

definition *fee-diff-on* **where**
 $fee-diff-on\ P\ P'\ I \longleftrightarrow id-grid-on\ P\ P'\ I \wedge (\forall i \in I. lq\ P\ i = lq\ P'\ i)$

lemma *fee-diff-onI*[*intro*]:
assumes *id-grid-on* $P\ P'\ I$
and $\bigwedge i. i \in I \implies lq\ P\ i = lq\ P'\ i$
shows *fee-diff-on* $P\ P'\ I$
 $\langle proof \rangle$

lemma *fee-diff-onD*[*dest*]:
assumes *fee-diff-on* $P\ P'\ I$
shows *id-grid-on* $P\ P'\ I \ \forall i \in I. lq\ P\ i = lq\ P'\ i$
 $\langle proof \rangle$

lemma *fee-diff-on-nz-liq*:
assumes *fee-diff-on* $P\ P'\ I$
shows *same-nz-liq-on* $P\ P'\ I$ $\langle proof \rangle$

lemma *fee-diff-on-comm*:
assumes *fee-diff-on* $P\ P'\ I$
shows *fee-diff-on* $P'\ P\ I$
 $\langle proof \rangle$

lemma *fee-diff-on-mono*:
assumes *fee-diff-on* $P\ P'\ I$

and $I' \subseteq I$
shows *fee-diff-on* $P \ P' \ I'$
 $\langle \text{proof} \rangle$

3 Grid refinement

We define the notion of pool refinement, that characterizes when a pool admits a finer price grid than another one but exhibits the same behavior.

3.1 Encompassement properties

definition *encomp-at* **where**

$\text{encomp-at } \gamma_1 \ \gamma_2 \ i \ k \equiv \gamma_2 \ k \leq \gamma_1 \ i \wedge$
 $\gamma_1 \ (i+1) \leq \gamma_2 \ (k+1)$

lemma *encomp-atD1*:

assumes $\text{encomp-at } \gamma_1 \ \gamma_2 \ i \ k$
shows $\gamma_2 \ k \leq \gamma_1 \ i$
 $\langle \text{proof} \rangle$

lemma *encomp-atD2*:

assumes $\text{encomp-at } \gamma_1 \ \gamma_2 \ i \ k$
shows $\gamma_1 \ (i+1) \leq \gamma_2 \ (k+1)$
 $\langle \text{proof} \rangle$

lemma *encomp-atI[intro]*:

assumes $\gamma_2 \ k \leq \gamma_1 \ i$
and $\gamma_1 \ (i+1) \leq \gamma_2 \ (k+1)$
shows $\text{encomp-at } \gamma_1 \ \gamma_2 \ i \ k \ \langle \text{proof} \rangle$

definition *encompassed* **where**

$\text{encompassed } \gamma_1 \ \gamma_2 \ k = \{i::\text{int}. \text{encomp-at } \gamma_1 \ \gamma_2 \ i \ k\}$

lemma *encompassed-convex*:

assumes $(i::\text{int}) \in \text{encompassed } \gamma_1 \ \gamma_2 \ k$
and $j \in \text{encompassed } \gamma_1 \ \gamma_2 \ k$
and $i \leq l$
and $l \leq j$
and *mono* γ_1
shows $l \in \text{encompassed } \gamma_1 \ \gamma_2 \ k \ \langle \text{proof} \rangle$

lemma *encompassed-interval*:

assumes *mono* γ_1
and *finite* $(\text{encompassed } \gamma_1 \ \gamma_2 \ k)$
and $\text{encompassed } \gamma_1 \ \gamma_2 \ k \neq \{\}$
shows $\text{encompassed } \gamma_1 \ \gamma_2 \ k =$
 $\{\text{Min } (\text{encompassed } \gamma_1 \ \gamma_2 \ k).. \text{Max } (\text{encompassed } \gamma_1 \ \gamma_2 \ k)\}$

$\langle proof \rangle$

lemma *encomp-at-idx-leq*:

fixes $\gamma_1::int \Rightarrow real$ **and** $\gamma_2::int \Rightarrow real$
assumes *strict-mono* ($\gamma_1::int \Rightarrow real$)
and *mono* ($\gamma_2::int \Rightarrow real$)
and *encomp-at* $\gamma_1 \gamma_2 i k$
and $\gamma_2 k' \leq \gamma_1 i$
shows $k' \leq k$
 $\langle proof \rangle$

lemma *encomp-at-unique*:

assumes *strict-mono* ($\gamma_1::int \Rightarrow real$)
and *mono* ($\gamma_2::int \Rightarrow real$)
and *encomp-at* $\gamma_1 \gamma_2 i k$
and *encomp-at* $\gamma_1 \gamma_2 i k'$
shows $k = k'$
 $\langle proof \rangle$

lemma *encomp-at-unique'*:

assumes *strict-mono* ($\gamma_1::int \Rightarrow real$)
and *mono* ($\gamma_2::int \Rightarrow real$)
and *encomp-at* $\gamma_1 \gamma_2 i k$
and $\gamma_2 k' \leq \gamma_1 i$
and $\gamma_1 i < \gamma_2 (k'+1)$
shows $k = k'$
 $\langle proof \rangle$

lemma *encomp-at-refl*:

fixes $\gamma::'a::\{one, plus\} \Rightarrow real$
shows *encomp-at* $\gamma \gamma i i$
 $\langle proof \rangle$

3.2 Finer price grids

definition *finer-range*:: $(int \Rightarrow real) \Rightarrow (int \Rightarrow real) \Rightarrow bool$ **where**
finer-range $\gamma_1 \gamma_2 \equiv (\forall i. \exists k. \text{encomp-at } \gamma_1 \gamma_2 i k)$

definition *finer-grid* **where**

finer-grid $P1 P2 \equiv \text{finer-range } (grd P1) (grd P2)$

lemma *finer-grid-range[simp]*:

assumes *finer-grid* $P1 P2$
shows *finer-range* $(grd P1) (grd P2)$
 $\langle proof \rangle$

definition *coarse-idx* **where**

coarse-idx $\gamma_1 \gamma_2 i =$
 $(THE k. \text{encomp-at } \gamma_1 \gamma_2 i k)$

definition *finer-idx-bound* **where**
finer-idx-bound $\gamma_1 \gamma_2 i =$
 (THE k . $\gamma_1 k = \gamma_2 (\text{coarse-idx } \gamma_1 \gamma_2 i)$)

lemma *finer-range-refl*:
 shows *finer-range* $\gamma \gamma$ $\langle \text{proof} \rangle$

locale *finer-ranges* =
 fixes $\gamma_1 :: \text{int} \Rightarrow \text{real}$ **and** $\gamma_2 :: \text{int} \Rightarrow \text{real}$
 assumes *stm*: *strict-mono* γ_1
 and *mon*: *mono* γ_2
 and *fin*: *finer-range* $\gamma_1 \gamma_2$

begin

lemma *encomp-idx-unique*:
 shows $\exists! k$. *encomp-at* $\gamma_1 \gamma_2 i k$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-bounds*:
 shows *encomp-at* $\gamma_1 \gamma_2 i (\text{coarse-idx } \gamma_1 \gamma_2 i)$
 $\langle \text{proof} \rangle$

lemma *encompassed-bounds*:
 shows $i \in \text{encompassed } \gamma_1 \gamma_2 (\text{coarse-idx } \gamma_1 \gamma_2 i)$
 $\langle \text{proof} \rangle$

lemma *encompassed-unique*:
 assumes $i \in \text{encompassed } \gamma_1 \gamma_2 k$
 shows $k = \text{coarse-idx } \gamma_1 \gamma_2 i$
 $\langle \text{proof} \rangle$

lemma *encompassed-inj*:
 assumes $k \neq k'$
 shows $\text{encompassed } \gamma_1 \gamma_2 k \cap \text{encompassed } \gamma_1 \gamma_2 k' = \{\}$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-eq*:
 assumes $\gamma_2 k' \leq \gamma_1 i$
 and $\gamma_1 i < \gamma_2 (k'+1)$
 shows $k' = \text{coarse-idx } \gamma_1 \gamma_2 i$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-reached*:
 assumes $\gamma_1 m \leq \gamma_2 k$
 and $\gamma_2 k < \gamma_1 M$
 and $k = \text{coarse-idx } \gamma_1 \gamma_2 i$

shows $\exists j. \text{gamma1 } j = \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-reached-unique*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k < \text{gamma1 } M$
and $k = \text{coarse-idx } \text{gamma1 } \text{gamma2 } i$
shows $\exists! j. \text{gamma1 } j = \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *encomp-idx-mono*:
assumes $i < j$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } i k$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } j l$
and $k \neq l$
shows $k < l$
 $\langle \text{proof} \rangle$

lemma *encomp-idx-mono'*:
assumes $i \leq j$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } i k$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } j l$
shows $k \leq l$
 $\langle \text{proof} \rangle$

lemma *encomp-idx-mono-conv*:
assumes $k < l$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } i k$
and $\text{encomp-at } \text{gamma1 } \text{gamma2 } j l$
shows $i < j$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-eq*:
assumes $\text{gamma1 } m \leq \text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i)$
and $\text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i) < \text{gamma1 } M$
shows $\text{gamma1 } (\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } i) =$
 $\text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i)$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-exists-eq*:
assumes $\exists m. \text{gamma1 } m \leq \text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i)$
and $\exists M. \text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i) < \text{gamma1 } M$
shows $\text{gamma1 } (\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } i) =$
 $\text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i) \langle \text{proof} \rangle$

lemma *finer-idx-bound-eq'*:
assumes $i \in \text{encompassed } \text{gamma1 } \text{gamma2 } k$
and $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k < \text{gamma1 } M$

shows $\text{gamma1} \text{ (finer-idx-bound gamma1 gamma2 i) = gamma2 k}$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-exists-eq'*:
assumes $i \in \text{encompassed gamma1 gamma2 k}$
and $\exists m. \text{gamma1 } m \leq \text{gamma2 } k$
and $\exists M. \text{gamma2 } k < \text{gamma1 } M$
shows $\text{gamma1} \text{ (finer-idx-bound gamma1 gamma2 i) = gamma2 k}$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-mem*:
assumes $\text{gamma1 } m \leq \text{gamma2} \text{ (coarse-idx gamma1 gamma2 i)}$
and $\text{gamma2} \text{ (coarse-idx gamma1 gamma2 i + 1) } \leq \text{gamma1 } M$
and $\text{gamma2} \text{ (coarse-idx gamma1 gamma2 i) } \neq$
 $\text{gamma2} \text{ (coarse-idx gamma1 gamma2 i + 1)}$
shows $\text{finer-idx-bound gamma1 gamma2 i} \in$
 $\text{encompassed gamma1 gamma2} \text{ (coarse-idx gamma1 gamma2 i)}$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-reached*:
assumes $\text{gamma1 } m \leq \text{gamma2} \text{ (coarse-idx gamma1 gamma2 i)}$
and $\text{gamma2} \text{ (coarse-idx gamma1 gamma2 i) } < \text{gamma1 } M$
and $\text{gamma1 } i = \text{gamma2} \text{ (coarse-idx gamma1 gamma2 i)}$
shows $i = \text{finer-idx-bound gamma1 gamma2 i}$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-leq*:
assumes $\text{gamma1 } m \leq \text{gamma2} \text{ (coarse-idx gamma1 gamma2 i)}$
and $\text{gamma2} \text{ (coarse-idx gamma1 gamma2 i) } < \text{gamma1 } M$
shows $\text{finer-idx-bound gamma1 gamma2 i} \leq i$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-proj*:
assumes $i \in \text{encompassed gamma1 gamma2 k}$
and $j \in \text{encompassed gamma1 gamma2 k}$
and $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k < \text{gamma1 } M$
shows $\text{finer-idx-bound gamma1 gamma2 i} = \text{finer-idx-bound gamma1 gamma2 j}$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-min*:
assumes $i \in \text{encompassed gamma1 gamma2 k}$
and $j \in \text{encompassed gamma1 gamma2 k}$
and $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k < \text{gamma1 } M$
shows $\text{finer-idx-bound gamma1 gamma2 i} \leq j$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-finer-bound*:

assumes $\text{gamma1 } m \leq \text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i)$
and $\text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i) < \text{gamma1 } M$
shows $\text{coarse-idx } \text{gamma1 } \text{gamma2 } (\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } i) =$
 $\text{coarse-idx } \text{gamma1 } \text{gamma2 } i$
 $\langle \text{proof} \rangle$

lemma *finer-idx-bound-invol*:
assumes $\text{gamma1 } m \leq \text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i)$
and $\text{gamma2 } (\text{coarse-idx } \text{gamma1 } \text{gamma2 } i) < \text{gamma1 } M$
shows $\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } (\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } i) =$
 $\text{finer-idx-bound } \text{gamma1 } \text{gamma2 } i$
 $\langle \text{proof} \rangle$

lemma *reached-imp-coarse*:
assumes $\text{gamma1 } i = \text{gamma2 } k$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{gamma1 } (i+1) \leq \text{gamma2 } (k+1)$
 $\langle \text{proof} \rangle$

lemma *less-imp-coarse*:
assumes $\text{gamma1 } m < \text{gamma2 } k$
and $\text{gamma2 } k \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\exists i. \text{encomp-at } \text{gamma1 } \text{gamma2 } i k$
 $\langle \text{proof} \rangle$

lemma *ex-coarse-rep*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\exists i. \text{encomp-at } \text{gamma1 } \text{gamma2 } i k$
 $\langle \text{proof} \rangle$

lemma *encompassed-ne*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } k \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{encompassed } \text{gamma1 } \text{gamma2 } k \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *encompassed-ne'*:
assumes $\exists m. \text{gamma1 } m \leq \text{gamma2 } k$
and $\exists M. \text{gamma2 } k \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{encompassed } \text{gamma1 } \text{gamma2 } k \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *encompassed-finite*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$

and $\text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{finite } (\text{encompassed } \text{gamma1 } \text{gamma2 } k)$
 $\langle \text{proof} \rangle$

lemma *encompassed-finite'*:
assumes $\exists m. \text{gamma1 } m \leq \text{gamma2 } k$
and $\exists M. \text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{finite } (\text{encompassed } \text{gamma1 } \text{gamma2 } k)$ $\langle \text{proof} \rangle$

lemma *encompassed-Min-in*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{Min } (\text{encompassed } \text{gamma1 } \text{gamma2 } k) \in \text{encompassed } \text{gamma1 } \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *encompassed-Max-in*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{Max } (\text{encompassed } \text{gamma1 } \text{gamma2 } k) \in \text{encompassed } \text{gamma1 } \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *encompassed-min-gamma-eq*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{gamma1 } (\text{Min } (\text{encompassed } \text{gamma1 } \text{gamma2 } k)) = \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *encompassed-min-gamma-eq'*:
assumes $\exists m. \text{gamma1 } m \leq \text{gamma2 } k$
and $\exists M. \text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
shows $\text{gamma1 } (\text{Min } (\text{encompassed } \text{gamma1 } \text{gamma2 } k)) = \text{gamma2 } k$
 $\langle \text{proof} \rangle$

lemma *coarse-idx-upper*:
assumes $\text{gamma2 } k < \text{gamma1 } j$
and $j \notin \text{encompassed } \text{gamma1 } \text{gamma2 } k$
shows $k < \text{coarse-idx } \text{gamma1 } \text{gamma2 } j$
 $\langle \text{proof} \rangle$

lemma *encompassed-max-Suc-eq*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } (k+1) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$

and $\text{gamma2 } (k+1) \neq \text{gamma2 } (k+2)$
shows $\text{Max } (\text{encompassed } \text{gamma1 } \text{gamma2 } k) + 1 \in$
 $\text{encompassed } \text{gamma1 } \text{gamma2 } (k+1)$
 $\langle \text{proof} \rangle$

lemma *encompassed-max-Suc-gamma-eq*:
assumes $\text{gamma1 } m \leq \text{gamma2 } k$
and $\text{gamma2 } (k+2) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
and $\text{gamma2 } (k+1) \neq \text{gamma2 } (k+2)$
shows $\text{gamma1 } (\text{Max } (\text{encompassed } \text{gamma1 } \text{gamma2 } k) + 1) = \text{gamma2 } (k+1)$
 $\langle \text{proof} \rangle$

lemma *encompassed-max-Suc-gamma-eq'*:
assumes $\exists m. \text{gamma1 } m \leq \text{gamma2 } k$
and $\exists M. \text{gamma2 } (k+2) \leq \text{gamma1 } M$
and $\text{gamma2 } k \neq \text{gamma2 } (k+1)$
and $\text{gamma2 } (k+1) \neq \text{gamma2 } (k+2)$
shows $\text{gamma1 } (\text{Max } (\text{encompassed } \text{gamma1 } \text{gamma2 } k) + 1) = \text{gamma2 } (k+1)$
 $\langle \text{proof} \rangle$

end

lemma *coarse-idx-refl*:
fixes $\text{gamma}::\text{int} \Rightarrow \text{real}$
assumes *strict-mono gamma*
shows $i = \text{coarse-idx } \text{gamma } \text{gamma } i$
 $\langle \text{proof} \rangle$

3.3 Pools with finer grids and coinciding profiles

definition *pool-coarse-idx* **where**
 $\text{pool-coarse-idx} = (\lambda P1 P2 i. \text{coarse-idx } (\text{grd } P1) (\text{grd } P2) i)$

lemma *pool-coarse-idxD*:
assumes $k = \text{pool-coarse-idx } P1 P2 i$
shows $k = \text{coarse-idx } (\text{grd } P1) (\text{grd } P2) i$
 $\langle \text{proof} \rangle$

definition *pool-finer-idx-bound* **where**
 $\text{pool-finer-idx-bound} = (\lambda P1 P2 i. \text{finer-idx-bound } (\text{grd } P1) (\text{grd } P2) i)$

lemma *pool-finer-idx-boundD*:
assumes $l = \text{pool-finer-idx-bound } P1 P2 i$
shows $l = \text{finer-idx-bound } (\text{grd } P1) (\text{grd } P2) i$
 $\langle \text{proof} \rangle$

definition *finer-pool* **where**
 $\text{finer-pool } P1 P2 \equiv \text{finer-grid } P1 P2 \wedge$

$(\forall i. lq\ P1\ i = lq\ P2\ (pool-coarse-idx\ P1\ P2\ i)) \wedge$
 $(\forall i. fee\ P1\ i = fee\ P2\ (pool-coarse-idx\ P1\ P2\ i))$

lemma *finer-poolI*[intro]:
assumes *finer-range* (*grd* *P1*) (*grd* *P2*)
and $(\forall i. lq\ P1\ i = lq\ P2\ (pool-coarse-idx\ P1\ P2\ i))$
and $(\forall i. fee\ P1\ i = fee\ P2\ (pool-coarse-idx\ P1\ P2\ i))$
shows *finer-pool* *P1* *P2*
 $\langle proof \rangle$

lemma *finer-poolD*:
assumes *finer-pool* *P1* *P2* **shows**
finer-range (*grd* *P1*) (*grd* *P2*)
 $\forall i. lq\ P1\ i = lq\ P2\ (pool-coarse-idx\ P1\ P2\ i)$
 $\forall i. fee\ P1\ i = fee\ P2\ (pool-coarse-idx\ P1\ P2\ i)$
 $\langle proof \rangle$

lemma *finer-pool-refl*:
assumes *strict-mono* (*grd* *P*)
shows *finer-pool* *P* *P*
 $\langle proof \rangle$

locale *finer-pools* =
fixes *P1* *P2*
assumes *fin-pool*: *finer-pool* *P1* *P2*

begin

lemma *finer-pool-grid*:
shows *finer-range* (*grd* *P1*) (*grd* *P2*) $\langle proof \rangle$

lemma *finer-pool-liq*:
shows $\forall i. lq\ P1\ i = lq\ P2\ (pool-coarse-idx\ P1\ P2\ i)$
 $\langle proof \rangle$

lemma *finer-pool-fee*:
shows $\forall i. fee\ P1\ i = fee\ P2\ (pool-coarse-idx\ P1\ P2\ i)$
 $\langle proof \rangle$

lemma *encompassed-liq-eq*:
assumes *strict-mono* (*grd* *P1*)
and *mono* (*grd* *P2*)
and $i \in encompassed\ (grd\ P1)\ (grd\ P2)\ k$
shows $lq\ P1\ i = lq\ P2\ k$
 $\langle proof \rangle$

lemma *encompassed-fee-eq*:
assumes *strict-mono* (*grd* *P1*)
and *mono* (*grd* *P2*)

and $i \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) k$
shows $\text{fee } P1 i = \text{fee } P2 k$
 $\langle \text{proof} \rangle$

lemma *sum-rng-token:*

assumes *strict-mono* ($\text{grd } P1$)
and *mono* ($\text{grd } P2$)
and $\text{grd } P1 m1 \leq \text{grd } P2 k$
and $\text{grd } P2 (k+1) \leq \text{grd } P1 M1$
and $\text{grd } P2 k \neq \text{grd } P2 (k + 1)$
and $\bigwedge a b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) b \implies$
 $g (\text{lq } P1) a = g' (\text{lq } P2) b$
and $\forall i \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) k. \text{dff } x i = f (i+1) - f i$
shows $\text{sum } (\text{rng-token dff } (g (\text{lq } P1)) x)$
 $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) k) =$
 $(g' (\text{lq } P2)) k * (f (\text{Max } (\text{encompassed } (\text{grd } P1) (\text{grd } P2) k) + 1) -$
 $f (\text{Min } (\text{encompassed } (\text{grd } P1) (\text{grd } P2) k)))$
 $\langle \text{proof} \rangle$

lemma *sum-rng-gen-quote:*

assumes *strict-mono* ($\text{grd } P1$)
and *mono* ($\text{grd } P2$)
and $\text{grd } P1 m1 \leq \text{grd } P2 k$
and $\text{grd } P2 (k+2) \leq \text{grd } P1 M1$
and $\text{grd } P2 k \neq \text{grd } P2 (k + 1)$
and $\text{grd } P2 (k+1) \neq \text{grd } P2 (k + 2)$
and $\bigwedge a b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) b \implies$
 $f (\text{lq } P1) a = f' (\text{lq } P2) b$
shows $\text{sum } (\text{rng-gen-quote } (\text{grd } P1) (f (\text{lq } P1)) x)$
 $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) k) =$
 $\text{rng-gen-quote } (\text{grd } P2) (f' (\text{lq } P2)) x k$
 $\langle \text{proof} \rangle$

lemma *sum-rng-gen-base:*

assumes *strict-mono* ($\text{grd } P1$)
and *mono* ($\text{grd } P2$)
and $\text{grd } P1 m1 \leq \text{grd } P2 k$
and $\text{grd } P2 (k+2) \leq \text{grd } P1 M1$
and $\text{grd } P2 k \neq \text{grd } P2 (k + 1)$
and $\text{grd } P2 (k+1) \neq \text{grd } P2 (k + 2)$
and $\bigwedge a b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) b \implies$
 $f (\text{lq } P1) a = f' (\text{lq } P2) b$
shows $\text{sum } (\text{rng-gen-base } (\text{grd } P1) (f (\text{lq } P1)) x)$
 $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) k) =$
 $\text{rng-gen-base } (\text{grd } P2) (f' (\text{lq } P2)) x k$
 $\langle \text{proof} \rangle$

lemma *finer-imp-finite-liq:*

assumes *strict-mono* ($\text{grd } P1$)

and *mono* (*grd* *P2*)
and *finite-liq* *P2*
and $\bigwedge k. \text{liq } P2 \ k \neq 0 \implies \text{finite } (\text{encompassed } (\text{grd } P1) (\text{grd } P2) \ k)$
shows *finite-liq* *P1*
 <proof>

lemma *finer-imp-finite-liq'*:
assumes *finer-pool* *P1* *P2*
and *strict-mono* (*grd* *P1*)
and *mono* (*grd* *P2*)
and *finite-liq* *P1*
and *finite* $\{k. \text{encompassed } (\text{grd } P1) (\text{grd } P2) \ k = \{\}\}$
shows *finite-liq* *P2*
 <proof>

end

3.4 Spanning grids

definition *span-grid* **where**
span-grid *P* $\longleftrightarrow \text{strict-mono } (\text{grd } P) \wedge (\forall i. 0 < \text{grd } P \ i) \wedge$
 $(\forall r > 0. \exists i. \text{grd } P \ i < r) \wedge (\forall r. \exists i. r < \text{grd } P \ i)$

lemma *span-gridD*:
assumes *span-grid* *P*
shows *strict-mono* (*grd* *P*) $\forall i. 0 < \text{grd } P \ i$
 $\forall r > 0. \exists i. \text{grd } P \ i < r \ \forall r. \exists i. r < \text{grd } P \ i$
 <proof>

lemma *span-gridI[intro]*:
assumes *strict-mono* (*grd* *P*)
and $\forall i. 0 < \text{grd } P \ i$
and $\forall r > 0. \exists i. \text{grd } P \ i < r$
and $\forall r. \exists i. r < \text{grd } P \ i$
shows *span-grid* *P* <proof>

lemma *span-grid-eq*:
assumes *span-grid* *P*
and *grd* *P* = *grd* *P'*
shows *span-grid* *P'* <proof>

locale *finer-spanning-pool* = *finer-pools* +
assumes *span*: *span-grid* *P1*

begin

lemma *finer-spanning-gt*:
shows $\exists i. r < \text{grd } P2 \ i$
 <proof>

lemma *finer-spanning-lt*:
 assumes $0 < r$
shows $\exists i. \text{grd } P2 \ i < r$
 $\langle \text{proof} \rangle$

lemma *finer-span-grid*:
 assumes $\forall i. 0 < \text{grd } P2 \ i$
 and *strict-mono* ($\text{grd } P2$)
shows *span-grid* $P2$
 $\langle \text{proof} \rangle$

end

locale *finer-two-spanning-pools* = *finer-spanning-pool* +
 assumes *span2*: *span-grid* $P2$

sublocale *finer-two-spanning-pools* $\subseteq$ *finer-ranges* $\text{grd } P1 \ \text{grd } P2$
 $\langle \text{proof} \rangle$

context *finer-two-spanning-pools*
begin

lemma *spanning-sum-rng-gen-quote*:
 assumes $\bigwedge a \ b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) \ b \implies$
 $f \ (lq \ P1) \ a = f' \ (lq \ P2) \ b$
shows $\text{sum } (\text{rng-gen-quote } (\text{grd } P1) (f \ (lq \ P1))) \ x$
 $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) \ k) =$
 $\text{rng-gen-quote } (\text{grd } P2) (f' \ (lq \ P2)) \ x \ k$
 $\langle \text{proof} \rangle$

lemma *spanning-sum-rng-gen-base*:
 assumes $\bigwedge a \ b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) \ b \implies$
 $f \ (lq \ P1) \ a = f' \ (lq \ P2) \ b$
shows $\text{sum } (\text{rng-gen-base } (\text{grd } P1) (f \ (lq \ P1))) \ x$
 $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) \ k) =$
 $\text{rng-gen-base } (\text{grd } P2) (f' \ (lq \ P2)) \ x \ k$
 $\langle \text{proof} \rangle$

lemma *span-grid-encompassed*:
shows *finite* $(\text{encompassed } (\text{grd } P1) (\text{grd } P2) \ k)$
 $\langle \text{proof} \rangle$

lemma *span-grids-finite-liq*:
 assumes *finite-liq* $P2$
shows *finite-liq* $P1$
 $\langle \text{proof} \rangle$

lemma *span-grids-ex-le*:

shows $\exists m. \text{grd } P1 \ m \leq \text{grd } P2 \ k$
 $\langle \text{proof} \rangle$

lemma *span-grids-ex-ge*:
shows $\exists M. \text{grd } P2 \ k \leq \text{grd } P1 \ M$
 $\langle \text{proof} \rangle$

lemma *span-grids-encompassed-ne*:
shows *encompassed* (*grd* *P1*) (*grd* *P2*) *k* $\neq \{\}$
 $\langle \text{proof} \rangle$

end

3.5 Spanning grids and finite liquidity

locale *finer-two-span-finite-liq* = *finer-two-spanning-pools* +
assumes *fin-liq*: *finite-liq* *P1*

sublocale *finer-two-span-finite-liq* $\subseteq$ *finite-liq-pool* *P1*
 $\langle \text{proof} \rangle$

lemma (**in** *finer-two-span-finite-liq*) *span-grids-finite-liq'*:
shows *finite-liq* *P2*
 $\langle \text{proof} \rangle$

sublocale *finer-two-span-finite-liq* $\subseteq$ *finite-liq-pool* *P2*
 $\langle \text{proof} \rangle$

context *finer-two-span-finite-liq*
begin

lemma *finer-pool-encompassed-Union*:
shows $(\bigcup (\text{encompassed } (\text{grd } P1) (\text{grd } P2) \{i. \text{liq } P2 \ i \neq 0\})) =$
 $\{i. \text{liq } P1 \ i \neq 0\}$
 $\langle \text{proof} \rangle$

lemma *spanning-finer-gen-quote-eq*:
assumes $\bigwedge a \ b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) \ b \implies$
 $f (\text{liq } P1) \ a = f' (\text{liq } P2) \ b$
and $\bigwedge i. \text{liq } P2 \ i = 0 \implies f' (\text{liq } P2) \ i = 0$
and $\bigwedge i. \text{liq } P1 \ i = 0 \implies f (\text{liq } P1) \ i = 0$
shows $\text{gen-quote } (\text{grd } P1) (f (\text{liq } P1)) \ x = \text{gen-quote } (\text{grd } P2) (f' (\text{liq } P2)) \ x$
 $\langle \text{proof} \rangle$

lemma *spanning-finer-gen-base-eq*:
assumes $\bigwedge a \ b. a \in \text{encompassed } (\text{grd } P1) (\text{grd } P2) \ b \implies$
 $f (\text{liq } P1) \ a = f' (\text{liq } P2) \ b$
and $\bigwedge i. \text{liq } P2 \ i = 0 \implies f' (\text{liq } P2) \ i = 0$
and $\bigwedge i. \text{liq } P1 \ i = 0 \implies f (\text{liq } P1) \ i = 0$

shows $\text{gen-base } (\text{grd } P1) (f (lq P1)) x = \text{gen-base } (\text{grd } P2) (f' (lq P2)) x$
 $\langle \text{proof} \rangle$

end

end

theory *CLMM-Description* **imports** *Grid-Information*

begin

4 CLMM description

Definition of CLMMs (Concentrated Liquidity Market Makers)

4.1 Preliminary results

definition *clmm-dsc* **where**

$\text{clmm-dsc } P \longleftrightarrow (\text{span-grid } P) \wedge (\text{finite-liq } P) \wedge (\forall i. 0 \leq lq P i) \wedge$
 $(\forall i. 0 \leq fee P i) \wedge (\forall i. fee P i < 1)$

lemma *clmm-dscI[intro]*:

assumes *span-grid* P

and *finite-liq* P

and $\forall i. 0 \leq lq P i$

and $\forall i. 0 \leq fee P i$

and $\forall i. fee P i < 1$

shows *clmm-dsc* $P \langle \text{proof} \rangle$

lemma *clmm-dsc-span-grid*:

assumes *clmm-dsc* P

shows *span-grid* $P \langle \text{proof} \rangle$

lemma *clmm-dsc-grid[simp]*:

assumes *clmm-dsc* P

shows *strict-mono* $(\text{grd } P) (\forall i. 0 < \text{grd } P i)$

$(\forall r > 0. \exists i. \text{grd } P i < r) (\forall r. \exists i. r < \text{grd } P i)$

$\langle \text{proof} \rangle$

lemma *clmm-dsc-grd-Suc*:

assumes *clmm-dsc* P

shows $\text{grd } P i < \text{grd } P (i+1) \langle \text{proof} \rangle$

lemma *clmm-dsc-grd-smono*:

assumes *clmm-dsc* P

and $i < j$
shows $\text{grd } P \ i < \text{grd } P \ j$ $\langle \text{proof} \rangle$

lemma *clmm-dsc-grd-mono*:
assumes *clmm-dsc* P
and $i \leq j$
shows $\text{grd } P \ i \leq \text{grd } P \ j$ $\langle \text{proof} \rangle$

lemma *clmm-dsc-liq*:
assumes *clmm-dsc* P
shows $\text{finite-liq } P \ 0 \leq \text{liq } P \ i$ $\langle \text{proof} \rangle$

lemma *clmm-dsc-fees*:
assumes *clmm-dsc* P
shows $(\forall i. \ 0 \leq \text{fee } P \ i) \wedge (\forall i. \ \text{fee } P \ i < 1)$ $\langle \text{proof} \rangle$

lemma *clmm-dsc-fees-neq-1*:
assumes *clmm-dsc* P
shows $\forall i. \ \text{fee } P \ i \neq 1$
 $\langle \text{proof} \rangle$

lemma *clmm-dsc-gross-liq*:
assumes *clmm-dsc* P
shows $\text{nz-support } (\text{gross-fct } (\text{liq } P) (\text{fee } P)) = \text{nz-support } (\text{liq } P)$
 $\langle \text{proof} \rangle$

lemma *clmm-dsc-gross-liq-zero-iff*:
assumes *clmm-dsc* P
shows $(\text{liq } P \ i = 0) \longleftrightarrow (\text{gross-fct } (\text{liq } P) (\text{fee } P) \ i = 0)$
 $\langle \text{proof} \rangle$

lemma *gross-liq-gt*:
assumes *clmm-dsc* P
and $\text{liq } P \ i \neq 0$
and $L = \text{gross-fct } (\text{liq } P) (\text{fee } P)$
shows $0 < L \ i$ $\langle \text{proof} \rangle$

lemma *gross-liq-ge*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (\text{liq } P) (\text{fee } P)$
shows $0 \leq L \ i$ $\langle \text{proof} \rangle$

lemma *rng-quote-net-ge*:
assumes *clmm-dsc* P
shows $0 \leq \text{liq } P \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)$
 $\langle \text{proof} \rangle$

lemma *rng-quote-gross-ge*:
assumes *clmm-dsc* P

and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
shows $0 \leq L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-pos*:
assumes *clmm-dsc* P
shows $0 \leq \text{quote-gross } P \ sqp \ \langle \text{proof} \rangle$

lemma *clmm-quote-gross-mono*:
assumes *clmm-dsc* P
shows *mono* $(\text{quote-gross } P)$
 $\langle \text{proof} \rangle$

lemma *quote-gross-imp-sqp-lt*:
assumes *clmm-dsc* P
and $\text{quote-gross } P \ sqp < \text{quote-gross } P \ sqp'$
shows $sqp < sqp'$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-net-mono*:
assumes *clmm-dsc* P
shows *mono* $(\text{quote-net } P)$
 $\langle \text{proof} \rangle$

lemma *clmm-base-gross-antimono*:
assumes *clmm-dsc* P
shows *antimono* $(\text{base-gross } P)$
 $\langle \text{proof} \rangle$

lemma *clmm-base-net-antimono*:
assumes *clmm-dsc* P
shows *antimono* $(\text{base-net } P)$
 $\langle \text{proof} \rangle$

lemma *liq-grd-min*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $0 < \text{grd-min } P \ \langle \text{proof} \rangle$

lemma *liq-grd-min-max*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{grd-min } P < \text{grd-max } P$
 $\langle \text{proof} \rangle$

definition *rng-blw* **where**
 $\text{rng-blw } P \ \text{prc} = \{i. \text{grd } P \ i \leq \text{prc}\}$

lemma *rng-blw-mem[simp]*:

assumes $i \in \text{rng-blw } P \text{ } \text{prc}$
shows $\text{grd } P \ i \leq \text{prc}$ $\langle \text{proof} \rangle$

lemma *rng-blw-bdd-above*:
assumes *clmm-dsc* P
shows *bdd-above* $(\text{rng-blw } P \text{ } \text{prc})$ $\langle \text{proof} \rangle$

lemma *rng-blw-ne*:
assumes *clmm-dsc* P
and $0 < \text{prc}$
shows $\text{rng-blw } P \text{ } \text{prc} \neq \{\}$
 $\langle \text{proof} \rangle$

definition *lower-tick* **where**
 $\text{lower-tick } P \text{ } \text{prc} = \text{Sup } (\text{rng-blw } P \text{ } \text{prc})$

lemma *grd-lower-tick-cong*:
assumes $\text{grd } P1 = \text{grd } P2$
shows $\text{lower-tick } P1 \text{ } \text{sqp} = \text{lower-tick } P2 \text{ } \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *lower-tick-mem*:
assumes *clmm-dsc* P
and $0 < \text{prc}$
shows $\text{lower-tick } P \text{ } \text{prc} \in \text{rng-blw } P \text{ } \text{prc}$ $\langle \text{proof} \rangle$

lemma *lower-tick-geq*:
assumes *clmm-dsc* P
and $0 < \text{prc}$
shows $\text{grd } P \ (\text{lower-tick } P \text{ } \text{prc}) \leq \text{prc}$
 $\langle \text{proof} \rangle$

lemma *lower-tick-geq'*:
assumes *clmm-dsc* P
and $i \in \text{rng-blw } P \text{ } \text{prc}$
shows $i \leq \text{lower-tick } P \text{ } \text{prc}$ $\langle \text{proof} \rangle$

lemma *lower-tick-ubound*:
assumes *clmm-dsc* P
and $i = \text{lower-tick } P \text{ } \text{prc}$
shows $\text{prc} < \text{grd } P \ (i+1)$
 $\langle \text{proof} \rangle$

lemma *lower-tick-lbound*:
assumes *clmm-dsc* P
and $0 < \text{prc}$
and $i = \text{lower-tick } P \text{ } \text{prc}$
shows $\text{grd } P \ i \leq \text{prc}$ $\langle \text{proof} \rangle$

lemma *lower-tick-lt*:
 assumes *clmm-dsc P*
 and $0 < sqp'$
 and $i = \text{lower-tick } P \ sqp$
 and $j = \text{lower-tick } P \ sqp'$
 and $i < j$
shows $sqp < sqp'$
 $\langle \text{proof} \rangle$

lemma *lower-tick-lt'*:
 assumes *clmm-dsc P*
 and $0 < sqp'$
 and $i = \text{lower-tick } P \ sqp$
 and $j = \text{lower-tick } P \ sqp'$
 and $sqp' < sqp$
 and $\text{grd } P \ i = sqp$
shows $j < i$
 $\langle \text{proof} \rangle$

lemma *lower-tick-mono*:
 assumes *clmm-dsc P*
 and $0 < sqp$
 and $i = \text{lower-tick } P \ sqp$
 and $j = \text{lower-tick } P \ sqp'$
 and $sqp \leq sqp'$
shows $i \leq j$
 $\langle \text{proof} \rangle$

lemma *lower-tick-eq*:
 assumes *clmm-dsc P*
 and $\text{grd } P \ i = sqp$
shows $\text{lower-tick } P \ sqp = i$
 $\langle \text{proof} \rangle$

lemma *lower-tick-charact*:
 assumes *clmm-dsc P*
 and $\text{grd } P \ i \leq sqp$
 and $sqp < \text{grd } P \ (i+1)$
shows $\text{lower-tick } P \ sqp = i$
 $\langle \text{proof} \rangle$

lemma *lower-tick-grd-min*:
 assumes *strict-mono (grd P)*
shows $\text{idx-min } (lq \ P) = \text{lower-tick } P \ (\text{grd-min } P)$
 $\langle \text{proof} \rangle$

lemma *lower-tick-grd-max*:
 assumes *strict-mono (grd P)*
shows $\text{idx-max } (lq \ P) + 1 = \text{lower-tick } P \ (\text{grd-max } P)$

$\langle \text{proof} \rangle$

lemma *grd-max-gt-if*:
 assumes *clmm-dsc* P
 and $i = \text{lower-tick } P \text{ sqp}$
 and $\text{lq } P \ i \neq 0$
 shows $\text{sqp} < \text{grd-max } P$
 $\langle \text{proof} \rangle$

4.2 Quote token addition and withdrawal in a CLMM

lemma (in *finite-nz-support*) *clmm-gen-quote-sum*:
 assumes *clmm-dsc* P
 and $0 < \text{sqp}$
 and $j = \text{lower-tick } P \text{ sqp}$
 shows $\text{gen-quote } (\text{grd } P) \ L \ \text{sqp} =$
 $L \ j * (\text{sqp} - \text{grd } P \ j) +$
 $\text{sum } (\lambda \ i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ \{i. L \ i \neq 0 \wedge i < j\}$
 $\langle \text{proof} \rangle$

lemma *clmm-gen-quote-grd-min*:
 assumes *clmm-dsc* P
 and $\text{nz-support } L \neq \{\}$
 and *finite* ($\text{nz-support } L$)
 and $\text{nz-support } L = \text{nz-support } (\text{lq } P)$
 shows $\text{gen-quote } (\text{grd } P) \ L \ (\text{grd-min } P) = 0 \ \langle \text{proof} \rangle$

lemma (in *finite-nz-support*) *clmm-gen-quote-grd-min-le*:
 assumes *clmm-dsc* P
 and $\text{nz-support } L = \text{nz-support } (\text{lq } P)$
 and $\text{sqp} \leq \text{grd-min } P$
 and $0 < \text{sqp}$
 shows $\text{gen-quote } (\text{grd } P) \ L \ \text{sqp} = 0$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-sum*:
 assumes *clmm-dsc* P
 and $0 < \text{sqp}$
 and $L = \text{gross-fct } (\text{lq } P) \ (\text{fee } P)$
 and $j = \text{lower-tick } P \ \text{sqp}$
 shows $\text{quote-gross } P \ \text{sqp} =$
 $L \ j * (\text{sqp} - \text{grd } P \ j) +$
 $\text{sum } (\lambda \ i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ \{i. L \ i \neq 0 \wedge i < j\}$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-grd-min*:
 assumes *clmm-dsc* P
 and $\text{nz-support } (\text{lq } P) \neq \{\}$
 shows $\text{quote-gross } P \ (\text{grd-min } P) = 0 \ \langle \text{proof} \rangle$

lemma *clmm-quote-gross-grd-min-le*:
 assumes *clmm-dsc* P
 and $sqp \leq \text{grd-min } P$
 and $0 < sqp$
shows $\text{quote-gross } P \text{ } sqp = 0$ $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-zero*:
 assumes *clmm-dsc* P
 and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-reach } P \ 0 = \text{grd-min } P$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-ge*:
 assumes *clmm-dsc* P
 and $\text{nz-support } (lq \ P) \neq \{\}$
 and $0 \leq y$
 and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
shows $\text{grd-min } P \leq (\text{quote-reach } P \ y)$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-pos*:
 assumes *clmm-dsc* P
 and $\text{nz-support } (lq \ P) \neq \{\}$
 and $0 \leq y$
 and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
 and $sqp = \text{quote-reach } P \ y$
shows $0 < sqp$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-mem*:
 assumes *clmm-dsc* P
 and $0 \leq y$
 and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
 and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-reach } P \ y \in \text{quote-gross } P - \{y\}$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-le*:
 assumes *clmm-dsc* P
 and $\text{nz-support } (lq \ P) \neq \{\}$
 and $0 < y$
 and $sqp \in \text{quote-gross } P - \{y\}$
 and $sqp' = \text{quote-reach } P \ y$
shows $sqp' \leq sqp$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-net-sum*:
 assumes *clmm-dsc* P

and $0 < sqp$
and $L = lq\ P$
and $j = \text{lower-tick}\ P\ sqp$
shows $\text{quote-net}\ P\ sqp =$

$$L\ j * (sqp - \text{grd}\ P\ j) +$$

$$\text{sum } (\lambda\ i. L\ i * (\text{grd}\ P\ (i+1) - \text{grd}\ P\ i))\ \{i. L\ i \neq 0 \wedge i < j\}$$
 $\langle \text{proof} \rangle$

lemma $\text{clmm-quote-net-grd-min}$:
assumes $\text{clmm-dsc}\ P$
and $\text{nz-support}\ (lq\ P) \neq \{\}$
shows $\text{quote-net}\ P\ (\text{grd-min}\ P) = 0\ \langle \text{proof} \rangle$

lemma $\text{clmm-quote-gross-reach-eq}$:
assumes $\text{clmm-dsc}\ P$
and $\text{nz-support}\ (lq\ P) \neq \{\}$
and $0 \leq y$
and $y \leq \text{quote-gross}\ P\ (\text{grd-max}\ P)$
shows $\text{quote-gross}\ P\ (\text{quote-reach}\ P\ y) = y$
 $\langle \text{proof} \rangle$

definition gen-quote-diff **where**
 $\text{gen-quote-diff}\ P\ L\ sqp\ sqp' = \text{gen-quote}\ (\text{grd}\ P)\ L\ sqp' - \text{gen-quote}\ (\text{grd}\ P)\ L\ sqp$

lemma (**in** finite-nz-support) $\text{clmm-gen-quote-diff-eq}$:
assumes $\text{clmm-dsc}\ P$
and $j = \text{lower-tick}\ P\ sqp$
and $k = \text{lower-tick}\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
shows $\text{gen-quote-diff}\ P\ L\ sqp\ sqp' = L\ k * (sqp' - \text{grd}\ P\ k) +$

$$\text{sum } (\lambda\ i. L\ i * (\text{grd}\ P\ (i+1) - \text{grd}\ P\ i))\ \{i. L\ i \neq 0 \wedge j < i \wedge i < k\} +$$

$$L\ j * (\text{grd}\ P\ (j+1) - sqp)$$
 $\langle \text{proof} \rangle$

lemma (**in** finite-nz-support) $\text{clmm-gen-quote-diff-eq'}$:
assumes $\text{clmm-dsc}\ P$
and $j = \text{lower-tick}\ P\ sqp$
and $j = \text{lower-tick}\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $L'\ j = L\ j$
shows $\text{gen-quote-diff}\ P\ L\ sqp\ sqp' = L'\ j * (sqp' - sqp)$
 $\langle \text{proof} \rangle$

lemma $\text{clmm-quote-gross-diff-eq}$:
assumes $\text{clmm-dsc}\ P$
and $L = \text{gross-fct}\ (lq\ P)\ (\text{fee}\ P)$

and $j = \text{lower-tick } P \text{ sqp}$
and $k = \text{lower-tick } P \text{ sqp}'$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
and $j < k$
shows $\text{quote-gross } P \text{ sqp}' - \text{quote-gross } P \text{ sqp} = L \ k * (\text{sqp}' - \text{grd } P \ k) +$
 $\text{sum } (\lambda \ i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ \{i. L \ i \neq 0 \wedge j < i \wedge i < k\} +$
 $L \ j * (\text{grd } P \ (j+1) - \text{sqp})$
 $\langle \text{proof} \rangle$

lemma *clmm-rng-quote-strict-pos*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $L \ i \neq 0$
shows $0 < L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i) \ \langle \text{proof} \rangle$

lemma *clmm-sum-rng-quote-pos*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
shows $0 \leq \text{sum } (\lambda \ i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ M$
 $\langle \text{proof} \rangle$

lemma *clmm-sum-rng-quote-strict-pos*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $L \ i \neq 0$
and $i \in M$
and *finite* M
shows $0 < \text{sum } (\lambda \ i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ M$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-eq-sum-only-if*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \ \text{sqp}$
and $k = \text{lower-tick } P \ \text{sqp}'$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
and $j < k$
and $\text{quote-gross } P \ \text{sqp}' = \text{quote-gross } P \ \text{sqp}$
and $j < i$
and $i < k$
shows $L \ i = 0$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-eq-sum-emp*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \ \text{sqp}$

and $k = \text{lower-tick } P \text{ } sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
and $\text{quote-gross } P \text{ } sqp' = \text{quote-gross } P \text{ } sqp$
shows $\{i. L \ i \neq 0 \wedge j < i \wedge i < k\} = \{\}$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-eq-lower-only-if*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \text{ } sqp$
and $k = \text{lower-tick } P \text{ } sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
and $\text{quote-gross } P \text{ } sqp' = \text{quote-gross } P \text{ } sqp$
shows $L \ j = 0$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-eq-upper-only-if*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \text{ } sqp$
and $k = \text{lower-tick } P \text{ } sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
and $\text{quote-gross } P \text{ } sqp' = \text{quote-gross } P \text{ } sqp$
shows $L \ k = 0 \vee \text{grd } P \ k = sqp'$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-diff-eq'*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \text{ } sqp$
and $j = \text{lower-tick } P \text{ } sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
shows $\text{quote-gross } P \text{ } sqp' - \text{quote-gross } P \text{ } sqp = L \ j * (sqp' - sqp)$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-gross-eq-lower-only-if'*:
assumes *clmm-dsc* P
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \text{ } sqp$
and $j = \text{lower-tick } P \text{ } sqp'$
and $0 < sqp$
and $sqp < sqp'$

and $\text{quote-gross } P \text{ sqp}' = \text{quote-gross } P \text{ sqp}$
shows $L j = 0$
 $\langle \text{proof} \rangle$

lemma *clmm-quote-reach-grd-liq*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $j = \text{lower-tick } P \ \text{sqp}$
and $\text{grd } P \ j = \text{sqp}$
and $\text{sqp} = \text{quote-reach } P \ y$
shows $lq \ P \ (j - 1) \neq 0$
 $\langle \text{proof} \rangle$

lemma *quote-gross-gt-grd-min*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{grd-min } P < \text{sqp}$
shows $0 < \text{quote-gross } P \ \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *quote-gross-pos-gt-grd-min*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < \text{quote-gross } P \ \text{sqp}$
shows $\text{grd-min } P < \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *quote-gross-disj-gt*:
assumes *clmm-dsc* P
and $i = \text{lower-tick } P \ \text{sqp}$
and $j = \text{lower-tick } P \ \text{sqp}'$
and $i \leq k$
and $k < j$
and $lq \ P \ k \neq 0$
and $0 < \text{sqp}$
and $0 < \text{sqp}'$
shows $\text{quote-gross } P \ \text{sqp} < \text{quote-gross } P \ \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *quote-gross-disj-gt'*:
assumes *clmm-dsc* P
and $i = \text{lower-tick } P \ \text{sqp}$
and $j = \text{lower-tick } P \ \text{sqp}'$
and $i < j$
and $lq \ P \ j \neq 0$
and $\text{grd } P \ j < \text{sqp}'$
and $0 < \text{sqp}$

and $0 < sqp'$
shows $quote-gross\ P\ sqp < quote-gross\ P\ sqp'$
 $\langle proof \rangle$

lemma *quote-gross-lower-eq-gt*:
assumes *clmm-dsc* P
and $j = lower-tick\ P\ sqp$
and $j = lower-tick\ P\ sqp'$
and $lq\ P\ j \neq 0$
and $0 < sqp$
and $sqp < sqp'$
shows $quote-gross\ P\ sqp < quote-gross\ P\ sqp'$
 $\langle proof \rangle$

lemma *quote-reach-gt-grd-min*:
assumes *clmm-dsc* P
and $nz-support\ (lq\ P) \neq \{\}$
and $0 < y$
and $y \leq quote-gross\ P\ (grd-max\ P)$
and $sqp = quote-reach\ P\ y$
shows $grd-min\ P < sqp$
 $\langle proof \rangle$

lemma *sqp-lt-grd-max-imp-idx*:
assumes *clmm-dsc* P
and $nz-support\ (lq\ P) \neq \{\}$
and $0 < sqp$
and $sqp < grd-max\ P$
and $i = lower-tick\ P\ sqp$
shows $i \leq idx-max\ (lq\ P)$
 $\langle proof \rangle$

lemma *quote-gross-lt-grd-max*:
assumes *clmm-dsc* P
and $nz-support\ (lq\ P) \neq \{\}$
and $0 < sqp$
and $sqp < grd-max\ P$
shows $quote-gross\ P\ sqp < quote-gross\ P\ (grd-max\ P)$
 $\langle proof \rangle$

lemma *idx-max-gt-liq*:
assumes *clmm-dsc* P
and $j = idx-max\ (lq\ P)$
shows $\forall k > j. lq\ P\ k = 0$
 $\langle proof \rangle$

lemma *idx-min-lt-liq*:
assumes *clmm-dsc* P
and $j = idx-min\ (lq\ P)$

shows $\forall k < j. \text{ lq } P \ k = 0$
 $\langle \text{proof} \rangle$

lemma *quote-reach-le'*:
assumes *clmm-dsc* P
and *grd-min* $P < \text{sqp}$
and $i = \text{lower-tick } P \ \text{sqp}$
and $\text{ lq } P \ i \neq 0$
and $y = \text{quote-gross } P \ \text{sqp}$
shows $\text{quote-reach } P \ y \leq \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *quote-reach-gross-le*:
assumes *clmm-dsc* P
and *grd-min* $P \leq \text{sqp}$
shows $\text{quote-reach } P \ (\text{quote-gross } P \ \text{sqp}) \leq \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *quote-reach-strict-mono*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{ lq } P) \neq \{\}$
and $0 \leq y1$
and $y1 < y2$
and $y2 \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $\text{sqp} = \text{quote-reach } P \ y1$
and $\text{sqp}' = \text{quote-reach } P \ y2$
shows $\text{sqp} < \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *quote-reach-mono*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{ lq } P) \neq \{\}$
and $0 \leq y1$
and $y1 \leq y2$
and $y2 \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $\text{sqp} = \text{quote-reach } P \ y1$
and $\text{sqp}' = \text{quote-reach } P \ y2$
shows $\text{sqp} \leq \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *grd-max-quote-reach*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{ lq } P) \neq \{\}$
shows $\text{quote-reach } P \ (\text{quote-gross } P \ (\text{grd-max } P)) = \text{grd-max } P$
 $\langle \text{proof} \rangle$

lemma *quote-reach-gt*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{ lq } P) \neq \{\}$

and $0 < y$
and $y + \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp})$
shows $\text{sqp} < \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *lt-quote-gross-imp-up-price*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 < y$
and $y \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{quote-gross } P \text{ sqp} < y$
and $\text{sqp}' = \text{quote-reach } P y$
shows $\text{sqp} < \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *quote-reach-add-gt*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 < y$
and $y + \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp})$
shows $\text{quote-gross } P \text{ sqp} < \text{quote-gross } P \text{ sqp}'$
 $\langle \text{proof} \rangle$

lemma *quote-reach-leq-grd-max*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 \leq y$
and $y \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp} = \text{quote-reach } P y$
shows $\text{sqp} \leq \text{grd-max } P$
 $\langle \text{proof} \rangle$

lemma *quote-gross-grd-max-ge*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $\text{grd-max } P < \text{sqp}$
shows $\text{quote-gross } P \text{ sqp} = \text{quote-gross } P (\text{grd-max } P)$
 $\langle \text{proof} \rangle$

lemma *quote-gross-grd-max-max*:
assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
shows $\text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
 $\langle \text{proof} \rangle$

lemma *gross-grd-max-max'*:
assumes *clmm-dsc* P

and $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{sqp} < \text{grd-max } P$
shows $\text{quote-gross } P \ \text{sqp} < \text{quote-gross } P \ (\text{grd-max } P)$
 $\langle \text{proof} \rangle$

lemma *quote-reach-le-gross*:
assumes $\text{clmm-dsc } P$
and $0 < y$
and $0 < \text{sqp}$
and $y \leq \text{quote-gross } P \ \text{sqp}$
and $\text{sqp} \leq \text{grd-max } P$
and $\text{sqp}' = \text{quote-reach } P \ y$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{sqp}' \leq \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *quote-net-diff-eq*:
assumes $\text{clmm-dsc } P$
and $L = lq \ P$
and $j = \text{lower-tick } P \ \text{sqp}$
and $k = \text{lower-tick } P \ \text{sqp}'$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
and $j < k$
shows $\text{quote-net } P \ \text{sqp}' - \text{quote-net } P \ \text{sqp} = L \ k * (\text{sqp}' - \text{grd } P \ k) +$
 $\text{sum } (\lambda i. L \ i * (\text{grd } P \ (i+1) - \text{grd } P \ i)) \ \{i. L \ i \neq 0 \wedge j < i \wedge i < k\} +$
 $L \ j * (\text{grd } P \ (j+1) - \text{sqp})$
 $\langle \text{proof} \rangle$

lemma *quote-net-diff-eq'*:
assumes $\text{clmm-dsc } P$
and $L = lq \ P$
and $j = \text{lower-tick } P \ \text{sqp}$
and $j = \text{lower-tick } P \ \text{sqp}'$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
shows $\text{quote-net } P \ \text{sqp}' - \text{quote-net } P \ \text{sqp} = L \ j * (\text{sqp}' - \text{sqp})$
 $\langle \text{proof} \rangle$

4.3 Base token addition and withdrawal in a CLMM

lemma (*in finite-nz-support*) *gen-base-sum*:
assumes $\text{clmm-dsc } P$
and $0 < \text{sqp}$
and $j = \text{lower-tick } P \ \text{sqp}$
shows $\text{gen-base } (\text{grd } P) \ L \ \text{sqp} =$
 $L \ j * (\text{inverse } \text{sqp} - \text{inverse } (\text{grd } P \ (j+1))) +$
 $\text{sum } (\lambda i. L \ i * (\text{inverse } (\text{grd } P \ i) - \text{inverse } (\text{grd } P \ (i+1))))$
 $\{i. L \ i \neq 0 \wedge j < i\}$

$\langle \text{proof} \rangle$

lemma (in *finite-nz-support*) *gen-base-grd-max*:

assumes *clmm-dsc* P

and *nz-support* $L \neq \{\}$

and *nz-support* $L = \text{nz-support } (lq \ P)$

shows *gen-base* (*grd* P) L (*grd-max* P) = 0

$\langle \text{proof} \rangle$

lemma *base-gross-sum*:

assumes *clmm-dsc* P

and $0 < sqp$

and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$

and $j = \text{lower-tick } P \ sqp$

shows *base-gross* $P \ sqp =$

$$L \ j * (\text{inverse } sqp - \text{inverse } (\text{grd } P \ (j+1))) + \\ \text{sum } (\lambda i. L \ i * (\text{inverse } (\text{grd } P \ i) - \text{inverse } (\text{grd } P \ (i+1)))) \\ \{i. L \ i \neq 0 \wedge j < i\}$$

$\langle \text{proof} \rangle$

lemma *clmm-base-gross-grd-max*:

assumes *clmm-dsc* P

and *nz-support* (*lq* P) $\neq \{\}$

shows *base-gross* P (*grd-max* P) = 0 $\langle \text{proof} \rangle$

lemma *liq-base-reach-max*:

assumes *clmm-dsc* P

and *nz-support* (*lq* P) $\neq \{\}$

shows *base-reach* $P \ 0 = \text{grd-max } P$

$\langle \text{proof} \rangle$

lemma *base-net-sum*:

assumes *clmm-dsc* P

and $0 < sqp$

and $L = lq \ P$

and $j = \text{lower-tick } P \ sqp$

shows *base-net* $P \ sqp =$

$$L \ j * (\text{inverse } sqp - \text{inverse } (\text{grd } P \ (j+1))) + \\ \text{sum } (\lambda i. L \ i * (\text{inverse } (\text{grd } P \ i) - \text{inverse } (\text{grd } P \ (i+1)))) \\ \{i. L \ i \neq 0 \wedge j < i\}$$

$\langle \text{proof} \rangle$

definition *gen-base-diff* **where**

gen-base-diff $P \ L \ sqp \ sqp' = \text{gen-base } (\text{grd } P) \ L \ sqp - \text{gen-base } (\text{grd } P) \ L \ sqp'$

lemma (in *finite-nz-support*) *gen-base-diff-eq*:

assumes *clmm-dsc* P

and $j = \text{lower-tick } P \ sqp$

and $k = \text{lower-tick } P \ sqp'$

and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
shows $gen-base-diff\ P\ L\ sqp\ sqp' =$
 $L\ j * (inverse\ sqp - inverse\ (grd\ P\ (j+1))) +$
 $sum\ (\lambda\ i.\ L\ i * (inverse\ (grd\ P\ i) - inverse\ (grd\ P\ (i+1))))$
 $\{i.\ L\ i \neq 0 \wedge j < i \wedge i < k\} +$
 $L\ k * (inverse\ (grd\ P\ k) - inverse\ sqp')$
 $\langle proof \rangle$

lemma $(in\ finite-nz-support)\ gen-base-diff-eq'$:
assumes $clmm-dsc\ P$
and $j = lower-tick\ P\ sqp$
and $j = lower-tick\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
shows $gen-base-diff\ P\ L\ sqp\ sqp' = L\ j * (inverse\ sqp - inverse\ sqp')$
 $\langle proof \rangle$

lemma $lower-tick-lt-grd-min$:
assumes $clmm-dsc\ P$
and $sqp < grd-min\ P$
and $0 < sqp$
and $j = lower-tick\ P\ sqp$
shows $j < idx-min\ (lq\ P)$
 $\langle proof \rangle$

lemma $(in\ finite-nz-support)\ gen-base-grd-min-le$:
assumes $clmm-dsc\ P$
and $nz-support\ L = nz-support\ (lq\ P)$
and $sqp < grd-min\ P$
and $0 < sqp$
shows $gen-base\ (grd\ P)\ L\ sqp = gen-base\ (grd\ P)\ L\ (grd-min\ P)$
 $\langle proof \rangle$

lemma $base-net-grd-min-lt$:
assumes $clmm-dsc\ P$
and $sqp < grd-min\ P$
and $0 < sqp$
shows $base-net\ P\ sqp = base-net\ P\ (grd-min\ P)$
 $\langle proof \rangle$

lemma $base-net-grd-min-le$:
assumes $clmm-dsc\ P$
and $sqp \leq grd-min\ P$
and $0 < sqp$
shows $base-net\ P\ sqp = base-net\ P\ (grd-min\ P)$
 $\langle proof \rangle$

lemma *base-gross-diff-eq*:
assumes *clmm-dsc P*
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \ sqp$
and $k = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
shows $\text{base-gross } P \ sqp - \text{base-gross } P \ sqp' =$
 $L \ j * (\text{inverse } sqp - \text{inverse } (\text{grd } P \ (j+1))) +$
 $\text{sum } (\lambda \ i. L \ i * (\text{inverse } (\text{grd } P \ i) - \text{inverse } (\text{grd } P \ (i+1))))$
 $\{i. L \ i \neq 0 \wedge j < i \wedge i < k\} +$
 $L \ k * (\text{inverse } (\text{grd } P \ k) - \text{inverse } sqp')$
 $\langle \text{proof} \rangle$

lemma *base-gross-diff-eq'*:
assumes *clmm-dsc P*
and $L = \text{gross-fct } (lq \ P) \ (\text{fee } P)$
and $j = \text{lower-tick } P \ sqp$
and $j = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
shows $\text{base-gross } P \ sqp - \text{base-gross } P \ sqp' = L \ j * (\text{inverse } sqp - \text{inverse } sqp')$
 $\langle \text{proof} \rangle$

lemma *base-net-diff-eq*:
assumes *clmm-dsc P*
and $L = lq \ P$
and $j = \text{lower-tick } P \ sqp$
and $k = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
shows $\text{base-net } P \ sqp - \text{base-net } P \ sqp' =$
 $L \ j * (\text{inverse } sqp - \text{inverse } (\text{grd } P \ (j+1))) +$
 $\text{sum } (\lambda \ i. L \ i * (\text{inverse } (\text{grd } P \ i) - \text{inverse } (\text{grd } P \ (i+1))))$
 $\{i. L \ i \neq 0 \wedge j < i \wedge i < k\} +$
 $L \ k * (\text{inverse } (\text{grd } P \ k) - \text{inverse } sqp')$
 $\langle \text{proof} \rangle$

lemma *base-net-diff-eq'*:
assumes *clmm-dsc P*
and $L = lq \ P$
and $j = \text{lower-tick } P \ sqp$
and $j = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
shows $\text{base-net } P \ sqp - \text{base-net } P \ sqp' = L \ j * (\text{inverse } sqp - \text{inverse } sqp')$
 $\langle \text{proof} \rangle$

lemma *cst-fee-base-gross-net-tick-eq*:
assumes *clmm-dsc P*
and $\bigwedge i. \text{fee } P \ i = \text{phi}$
and $j = \text{lower-tick } P \ sqp$
and $j = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
shows $\text{base-net } P \ sqp - \text{base-net } P \ sqp' =$
 $(1 - \text{phi}) * (\text{base-gross } P \ sqp - \text{base-gross } P \ sqp')$
 $\langle \text{proof} \rangle$

lemma *cst-fee-base-gross-net-tick-lt*:
assumes *clmm-dsc P*
and $\bigwedge i. \text{fee } P \ i = \text{phi}$
and $j = \text{lower-tick } P \ sqp$
and $k = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
shows $\text{base-net } P \ sqp - \text{base-net } P \ sqp' =$
 $(1 - \text{phi}) * (\text{base-gross } P \ sqp - \text{base-gross } P \ sqp')$
 $\langle \text{proof} \rangle$

lemma *cst-fee-base-gross-net*:
assumes *clmm-dsc P*
and $\bigwedge i. \text{fee } P \ i = \text{phi}$
and $0 < sqp$
and $sqp \leq sqp'$
shows $\text{base-net } P \ sqp - \text{base-net } P \ sqp' =$
 $(1 - \text{phi}) * (\text{base-gross } P \ sqp - \text{base-gross } P \ sqp')$
 $\langle \text{proof} \rangle$

lemma *base-net-eq-only-if*:
assumes *clmm-dsc P*
and $j = \text{lower-tick } P \ sqp$
and $k = \text{lower-tick } P \ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
and $\text{quote-gross } P \ sqp' = \text{quote-gross } P \ sqp$
shows $\text{base-net } P \ sqp' = \text{base-net } P \ sqp$
 $\langle \text{proof} \rangle$

lemma *base-net-eq-only-if'*:
assumes *clmm-dsc P*
and $L = \text{lq } P$
and $j = \text{lower-tick } P \ sqp$
and $j = \text{lower-tick } P \ sqp'$

and $0 < sqp$
and $sqp \leq sqp'$
and $quote-gross\ P\ sqp = quote-gross\ P\ sqp'$
shows $base-net\ P\ sqp = base-net\ P\ sqp'$
 $\langle proof \rangle$

lemma *quote-gross-equiv-base-net*:
assumes *clmm-dsc* P
and $0 < sqp$
and $sqp \leq sqp'$
and $quote-gross\ P\ sqp = quote-gross\ P\ sqp'$
shows $base-net\ P\ sqp = base-net\ P\ sqp'$
 $\langle proof \rangle$

lemma *quote-gross-equiv-base-net'*:
assumes *clmm-dsc* P
and $0 < sqp$
and $0 < sqp'$
and $quote-gross\ P\ sqp = quote-gross\ P\ sqp'$
shows $base-net\ P\ sqp = base-net\ P\ sqp'$
 $\langle proof \rangle$

lemma (*in finite-nz-support*) *gen-quote-le-badd*:
assumes *clmm-dsc* P
and $\bigwedge i. 0 \leq L\ i$
and $0 < sqp$
and $sqp \leq sqp'$
shows $gen-quote-diff\ P\ L\ sqp\ sqp' / (sqp' * sqp') \leq gen-base-diff\ P\ L\ sqp\ sqp'$
 $\langle proof \rangle$

lemma (*in finite-nz-support*) *gen-base-le-qadd*:
assumes *clmm-dsc* P
and $\bigwedge i. 0 \leq L\ i$
and $0 < sqp$
and $sqp \leq sqp'$
shows $gen-base-diff\ P\ L\ sqp\ sqp' \leq gen-quote-diff\ P\ L\ sqp\ sqp' / (sqp * sqp)$
 $\langle proof \rangle$

lemma *quote-swap-grd-min-zero*:
assumes *clmm-dsc* P
and $nz-support\ (lq\ P) \neq \{\}$
and $grd-min\ P \leq sqp$
and $sqp \leq grd-max\ P$
shows $quote-swap\ P\ sqp\ 0 = 0$
 $\langle proof \rangle$

lemma *quote-swap-zero*:
assumes *clmm-dsc* P
and $nz-support\ (lq\ P) \neq \{\}$

and $0 < sqp$
and $sqp \leq \text{grd-max } P$
shows $\text{quote-swap } P \text{ } sqp \text{ } 0 = 0$
 $\langle \text{proof} \rangle$

lemma *quote-swap-grd-min-zero'*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{grd-min } P \leq sqp$
and $\text{quote-gross } P \text{ } sqp \leq \text{quote-gross } P \text{ } (\text{grd-max } P)$
shows $\text{quote-swap } P \text{ } sqp \text{ } 0 = 0$
 $\langle \text{proof} \rangle$

lemma *quote-swap-zero'*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < sqp$
and $\text{quote-gross } P \text{ } sqp \leq \text{quote-gross } P \text{ } (\text{grd-max } P)$
shows $\text{quote-swap } P \text{ } sqp \text{ } 0 = 0$
 $\langle \text{proof} \rangle$

lemma *quote-swap-grd-min*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $sqp < \text{grd-min } P$
and $0 < sqp$
shows $\text{quote-swap } P \text{ } sqp \text{ } y = \text{quote-swap } P \text{ } (\text{grd-min } P) \text{ } y$
 $\langle \text{proof} \rangle$

lemma *quote-reach-gross-base-net*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < \text{quote-gross } P \text{ } sqp$
and $sqp' = \text{quote-reach } P \text{ } (\text{quote-gross } P \text{ } sqp)$
shows $\text{base-net } P \text{ } sqp' = \text{base-net } P \text{ } sqp$
 $\langle \text{proof} \rangle$

lemma *quote-reach-base-net*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < sqp$
and $sqp' = \text{quote-reach } P \text{ } (\text{quote-gross } P \text{ } sqp)$
shows $\text{base-net } P \text{ } sqp' = \text{base-net } P \text{ } sqp$
 $\langle \text{proof} \rangle$

lemma *base-le-quote-gross*:
assumes *clmm-dsc* P'
and $0 < sqp$
and $sqp \leq sqp'$

shows $\text{base-gross } P' \text{ sqp} - \text{base-gross } P' \text{ sqp}' \leq$
 $(\text{quote-gross } P' \text{ sqp}' - \text{quote-gross } P' \text{ sqp}) / (\text{sqp} * \text{sqp})$
 $\langle \text{proof} \rangle$

lemma *quote-le-base-gross*:
assumes *clmm-dsc* P'
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
shows $(\text{quote-gross } P' \text{ sqp}' - \text{quote-gross } P' \text{ sqp}) / (\text{sqp}' * \text{sqp}') \leq$
 $\text{base-gross } P' \text{ sqp} - \text{base-gross } P' \text{ sqp}'$
 $\langle \text{proof} \rangle$

lemma *base-net-quote-ubound*:
assumes *clmm-dsc* P'
and $\bigwedge i. \text{fee } P' i = \text{phi}$
and $\text{phi} < 1$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
shows $\text{base-net } P' \text{ sqp} - \text{base-net } P' \text{ sqp}' \leq$
 $(1 - \text{phi}) * (\text{quote-gross } P' \text{ sqp}' - \text{quote-gross } P' \text{ sqp}) / (\text{sqp} * \text{sqp})$
 $\langle \text{proof} \rangle$

lemma *base-net-quote-lbound*:
assumes *clmm-dsc* P'
and $\bigwedge i. \text{fee } P' i = \text{phi}$
and $0 < \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
shows $(1 - \text{phi}) * (\text{quote-gross } P' \text{ sqp}' - \text{quote-gross } P' \text{ sqp}) / (\text{sqp}' * \text{sqp}') \leq$
 $\text{base-net } P' \text{ sqp} - \text{base-net } P' \text{ sqp}'$
 $\langle \text{proof} \rangle$

4.4 Market depth and slippage for finer CLMMs

4.4.1 Finer pools

locale *finer-clmm* =
fixes $P1 \ P2$
assumes *abs1*: *clmm-dsc* $P1$ **and** *abs2*: *clmm-dsc* $P2$
and *finer*: *finer-pool* $P1 \ P2$

sublocale *finer-clmm* $\subseteq$ *finer-two-span-finite-liq*
 $\langle \text{proof} \rangle$

context *finer-clmm*
begin

lemma *finer-base-net-eq*:
shows $\text{base-net } P1 = \text{base-net } P2$
 $\langle \text{proof} \rangle$

lemma *finer-quote-net-eq*:
shows *quote-net P1 = quote-net P2*
 ⟨*proof*⟩

lemma *finer-base-gross-eq*:
shows *base-gross P1 = base-gross P2*
 ⟨*proof*⟩

lemma *finer-quote-gross-eq*:
shows *quote-gross P1 = quote-gross P2*
 ⟨*proof*⟩

lemma *finer-mkt-depth*:
shows *mkt-depth P1 = mkt-depth P2*
 ⟨*proof*⟩

end

4.4.2 Finer CLMMs with nonzero liquidity

locale *finer-clmm-ne* = *finer-clmm* +
assumes *nonempty-liq*: *nz-support (lq P1) ≠ {}*

context *finer-clmm-ne*
begin

lemma *id-max-Max-eq*:
assumes *i1 = idx-max (lq P1)*
and *k2 = pool-coarse-idx P1 P2 i1*
shows *i1 = Max (encompassed (grd P1) (grd P2) k2)*
 ⟨*proof*⟩

lemma *id-min-Min-eq*:
assumes *i1 = idx-min (lq P1)*
and *k2 = pool-coarse-idx P1 P2 i1*
shows *i1 = Min (encompassed (grd P1) (grd P2) k2)*
 ⟨*proof*⟩

lemma *idx-max-Suc-grd-eq*:
assumes *i1 = idx-max (lq P1)*
and *k2 = pool-coarse-idx P1 P2 i1*
shows *grd P1 (i1 + 1) = grd P2 (k2 + 1)*
 ⟨*proof*⟩

lemma *idx-min-grd-eq*:
assumes *i1 = idx-min (lq P1)*
and *k2 = pool-coarse-idx P1 P2 i1*
shows *grd P1 i1 = grd P2 k2*
 ⟨*proof*⟩

```

lemma abs-finer-idx-max-coarse:
  assumes clmm-dsc P1
  and clmm-dsc P2
  and finer-pool P1 P2
  and nz-support (lq P1) ≠ {}
  and i1 = idx-max (lq P1)
  and k2 = pool-coarse-idx P1 P2 i1
shows k2 = idx-max (lq P2)
  <proof>

lemma abs-finer-idx-min-coarse:
  assumes i1 = idx-min (lq P1)
  and k2 = pool-coarse-idx P1 P2 i1
shows k2 = idx-min (lq P2)
  <proof>

lemma abs-finer-idx-max-img-eq:
shows grd-max P1 = grd-max P2
  <proof>

lemma abs-finer-idx-min-img-eq:
shows grd-min P1 = grd-min P2
  <proof>

lemma finer-base-reach-eq:
shows base-reach P1 = base-reach P2 <proof>

lemma finer-quote-reach-eq:
shows quote-reach P1 = quote-reach P2 <proof>

lemma finer-base-slippage:
shows base-slippage P1 = base-slippage P2
  <proof>

lemma finer-quote-slippage:
shows quote-slippage P1 = quote-slippage P2
  <proof>

end

```

5 Inequalities related to fees

```

context finite-liq-pool
begin

```

```

lemma gross-fct-le:
  assumes  $0 \leq f\ i$ 
  and  $\phi\ i \leq \phi'\ i$ 

```

and $\text{phi}'\ i < 1$
shows $\text{gross-fct } f\ \text{phi}\ i \leq \text{gross-fct } f\ \text{phi}'\ i$
 $\langle \text{proof} \rangle$

lemma *gross-fct-lt*:
assumes $0 < f\ i$
and $\text{phi}\ i < \text{phi}'\ i$
and $\text{phi}'\ i < 1$
shows $\text{gross-fct } f\ \text{phi}\ i < \text{gross-fct } f\ \text{phi}'\ i$
 $\langle \text{proof} \rangle$

lemma *fee-diff-same-base-net*:
assumes $\text{clmm-dsc } P$
and $\text{clmm-dsc } P'$
and $I = \{k. L\ k \neq 0 \wedge j \leq k\}$
and $\text{fee-diff-on } P\ P'\ I$
and $\text{same-nz-liq-on } P\ P'\ \{k. j \leq k\}$
and $0 < \text{sqp}$
and $j = \text{lower-tick } P\ \text{sqp}$
and $L = \text{lq } P$
and $\text{lower-tick } P\ \text{sqp} = \text{lower-tick } P'\ \text{sqp}$
shows $\text{base-net } P\ \text{sqp} = \text{base-net } P'\ \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *fee-diff-le-imp-quote-gross*:
assumes $\text{clmm-dsc } P$
and $\text{clmm-dsc } P'$
and $\{k. L\ k \neq 0 \wedge k \leq j\} \subseteq I$
and $\text{fee-diff-on } P\ P'\ I$
and $\text{same-nz-liq-on } P\ P'\ \{k. k \leq j\}$
and $0 < \text{sqp}$
and $j = \text{lower-tick } P\ \text{sqp}$
and $L = \text{gross-fct } (\text{lq } P)\ (\text{fee } P)$
and $\bigwedge i. i \in I \implies \text{fee } P\ i \leq \text{fee } P'\ i$
and $\text{lower-tick } P\ \text{sqp} = \text{lower-tick } P'\ \text{sqp}$
shows $\text{quote-gross } P\ \text{sqp} \leq \text{quote-gross } P'\ \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *fee-diff-le-imp-quote-gross-mono*:
assumes $\text{clmm-dsc } P$
and $\text{clmm-dsc } P'$
and $\{k. L\ k \neq 0 \wedge k \leq j\} \subseteq I$
and $\text{fee-diff-on } P\ P'\ I$
and $\text{same-nz-liq-on } P\ P'\ \{k. k \leq j\}$
and $0 < \text{sqp}$
and $j = \text{lower-tick } P\ \text{sqp}$
and $L = \text{gross-fct } (\text{lq } P)\ (\text{fee } P)$
and $\bigwedge i. i \in I \implies \text{fee } P\ i \leq \text{fee } P'\ i$
and $\text{lower-tick } P\ \text{sqp} = \text{lower-tick } P'\ \text{sqp}$

and $sqp \leq sqp'$
shows $quote-gross\ P\ sqp \leq quote-gross\ P'\ sqp'$
 $\langle proof \rangle$

lemma *fee-diff-quote-diff-expand*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $L = gross-fct\ (lq\ P)\ (fee\ P)$
and $j = lower-tick\ P\ sqp$
and $k = lower-tick\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $j < k$
and $\{m. L\ m \neq 0 \wedge j \leq m \wedge m \leq k\} \subseteq I$
and *fee-diff-on* $P\ P'\ I$
and *same-nz-liq-on* $P\ P'\ \{m. j \leq m \wedge m \leq k+1\}$
and $\bigwedge i. i \in I \implies fee\ P\ i \leq fee\ P'\ i$
and $lower-tick\ P\ sqp = lower-tick\ P'\ sqp$
and $lower-tick\ P\ sqp' = lower-tick\ P'\ sqp'$
shows $quote-gross\ P\ sqp' - quote-gross\ P\ sqp \leq quote-gross\ P'\ sqp' - quote-gross\ P'\ sqp$
 $\langle proof \rangle$

lemma *fee-diff-quote-diff-expand'*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $L = gross-fct\ (lq\ P)\ (fee\ P)$
and $j = lower-tick\ P\ sqp$
and $j = lower-tick\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $L\ j \neq 0 \longrightarrow j \in I$
and *fee-diff-on* $P\ P'\ I$
and $\bigwedge i. i \in I \implies fee\ P\ i \leq fee\ P'\ i$
and $lower-tick\ P\ sqp = lower-tick\ P'\ sqp$
and $lower-tick\ P\ sqp' = lower-tick\ P'\ sqp'$
shows $quote-gross\ P\ sqp' - quote-gross\ P\ sqp \leq quote-gross\ P'\ sqp' - quote-gross\ P'\ sqp$
 $\langle proof \rangle$

lemma *fee-diff-quote-diff-le*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $L = gross-fct\ (lq\ P)\ (fee\ P)$
and $j = lower-tick\ P\ sqp$
and $k = lower-tick\ P\ sqp'$
and $0 < sqp$
and $sqp \leq sqp'$
and $\{m. L\ m \neq 0 \wedge j \leq m \wedge m \leq k\} \subseteq I$

and *fee-diff-on* $P P' I$
and *same-nz-liq-on* $P P' \{m. j \leq m \wedge m \leq k+1\}$
and $\bigwedge i. i \in I \implies \text{fee } P \ i \leq \text{fee } P' \ i$
and *lower-tick* $P \text{ sqp} = \text{lower-tick } P' \text{ sqp}$
and *lower-tick* $P \text{ sqp}' = \text{lower-tick } P' \text{ sqp}'$
shows $\text{quote-gross } P \text{ sqp}' - \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P' \text{ sqp}' - \text{quote-gross } P' \text{ sqp}$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-nz-support*:
assumes $i \in I$
and $\text{lq } P \ i \neq 0$
and *same-nz-liq-on* $P P' I$
shows $\text{nz-support } (\text{lq } P') \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-idx-max*:
assumes *finite-liq* P'
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $I = \{\text{idx-min } (\text{lq } P) .. \text{idx-max } (\text{lq } P) + 1\}$
and *same-nz-liq-on* $P P' I$
shows $\text{idx-max } (\text{lq } P) \leq \text{idx-max } (\text{lq } P')$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-grd-max*:
assumes *finite-liq* P'
and *mono* $(\text{grd } P')$
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $I = \{\text{idx-min } (\text{lq } P) .. \text{idx-max } (\text{lq } P) + 1\}$
and *same-nz-liq-on* $P P' I$
shows $\text{grd-max } P \leq \text{grd-max } P'$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-lower-tick*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and *same-nz-liq-on* $P P' \{i. i \leq j+1\}$
and $0 < \text{sqp}$
and *lower-tick* $P \text{ sqp} \leq j$
shows *lower-tick* $P' \text{ sqp} = \text{lower-tick } P \text{ sqp}$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-lower-tick'*:
assumes *clmm-dsc* P'
and *same-nz-liq-on* $P P' \{i. i \leq j\}$
and $\text{grd } P \ j = \text{sqp}$
shows *lower-tick* $P' \text{ sqp} = j$
 $\langle \text{proof} \rangle$

lemma *fee-diff-le-grd-max*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $\text{nz-support } (lq\ P) \neq \{\}$
and $\{\text{idx-min } (lq\ P) .. \text{idx-max } (lq\ P) + 1\} \subseteq I$
and *fee-diff-on* $P\ P'\ I$
and *same-nz-liq-on* $P\ P'\ \{k. k \leq \text{idx-max } (lq\ P) + 1\}$
and $\bigwedge i. i \in I \implies \text{fee } P\ i \leq \text{fee } P'\ i$
shows $\text{quote-gross } P\ (\text{grd-max } P) \leq \text{quote-gross } P'\ (\text{grd-max } P)$
 $\langle \text{proof} \rangle$

lemma *fee-diff-le-grd-max'*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $\text{nz-support } (lq\ P) \neq \{\}$
and $\{\text{idx-min } (lq\ P) .. \text{idx-max } (lq\ P) + 1\} \subseteq I$
and *fee-diff-on* $P\ P'\ I$
and *same-nz-liq-on* $P\ P'\ \{k. k \leq \text{idx-max } (lq\ P) + 1\}$
and $\bigwedge i. i \in I \implies \text{fee } P\ i \leq \text{fee } P'\ i$
shows $\text{quote-gross } P\ (\text{grd-max } P) \leq \text{quote-gross } P'\ (\text{grd-max } P')$
 $\langle \text{proof} \rangle$

lemma *fee-diff-le-imp-quote-reach*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $\text{nz-support } (lq\ P) \neq \{\}$
and $\{\text{idx-min } (lq\ P) .. \text{idx-max } (lq\ P) + 1\} \subseteq I$
and *fee-diff-on* $P\ P'\ I$
and *same-nz-liq-on* $P\ P'\ \{i. i \leq \text{idx-max } (lq\ P) + 1\}$
and $\bigwedge i. i \in I \implies \text{fee } P\ i \leq \text{fee } P'\ i$
and $0 < y$
and $y \leq \text{quote-gross } P\ (\text{grd-max } P)$
shows $\text{quote-reach } P'\ y \leq \text{quote-reach } P\ y$
 $\langle \text{proof} \rangle$

lemma *same-nz-liq-on-if-simil*:
assumes $\text{grd } P = \text{grd } P'$
and $\text{nz-support } (lq\ P) = \text{nz-support } (lq\ P')$
shows *same-nz-liq-on* $P\ P'\ I$
 $\langle \text{proof} \rangle$

lemma *fee-diff-simil-base-net*:
assumes *clmm-dsc* P
and *clmm-dsc* P'
and $\text{nz-support } (lq\ P) \neq \{\}$
and $\{\text{idx-min } (lq\ P) .. \text{idx-max } (lq\ P) + 1\} \subseteq I$
and *fee-diff-on* $P\ P'\ I$
and $\text{nz-support } (lq\ P) = \text{nz-support } (lq\ P')$
and $\text{grd } P = \text{grd } P'$

and $\text{grd-min } P \leq \text{sqp}$
 and $\text{sqp} \leq \text{grd-max } P$
 shows $\text{base-net } P \text{ sqp} = \text{base-net } P' \text{ sqp}$
 <proof>

lemma *fee-diff-le-price-cmp*:
 assumes *clmm-dsc* P
 and *clmm-dsc* P'
 and $\text{nz-support } (\text{lq } P) \neq \{\}$
 and $\{\text{idx-min } (\text{lq } P) .. \text{idx-max } (\text{lq } P) + 1\} \subseteq I$
 and *fee-diff-on* $P P' I$
 and $\text{nz-support } (\text{lq } P) = \text{nz-support } (\text{lq } P')$
 and $\text{grd } P = \text{grd } P'$
 and $\bigwedge i. i \in I \implies \text{fee } P i \leq \text{fee } P' i$
 and $0 < y$
 and $y + \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
 and $\text{grd-min } P \leq \text{sqp}$
 and $\text{sqp1} = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp})$
 and $\text{sqp2} = \text{quote-reach } P' (y + \text{quote-gross } P' \text{ sqp})$
 shows $\text{sqp2} \leq \text{sqp1}$
 <proof>

lemma *fee-diff-le-imp-quote-swap*:
 assumes *clmm-dsc* P
 and *clmm-dsc* P'
 and $\text{nz-support } (\text{lq } P) \neq \{\}$
 and $\{\text{idx-min } (\text{lq } P) .. \text{idx-max } (\text{lq } P) + 1\} \subseteq I$
 and *fee-diff-on* $P P' I$
 and $\text{nz-support } (\text{lq } P) = \text{nz-support } (\text{lq } P')$
 and $\text{grd } P = \text{grd } P'$
 and $\bigwedge i. i \in I \implies \text{fee } P i \leq \text{fee } P' i$
 and $0 < y$
 and $y + \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
 and $\text{grd-min } P \leq \text{sqp}$
 shows $\text{quote-swap } P' \text{ sqp } y \leq \text{quote-swap } P \text{ sqp } y$
 <proof>

lemma *fee-ge-quote-swap-le*:
 assumes *clmm-dsc* P
 and *clmm-dsc* P'
 and $\text{nz-support } (\text{lq } P) \neq \{\}$
 and $\text{grd } P = \text{grd } P'$
 and $\text{lq } P = \text{lq } P'$
 and $\bigwedge i. \text{fee } P i \leq \text{fee } P' i$
 and $0 \leq y$
 and $0 < \text{sqp}$
 and $y + \text{quote-gross } P \text{ sqp} \leq \text{quote-gross } P (\text{grd-max } P)$
 shows $\text{quote-swap } P' \text{ sqp } y \leq \text{quote-swap } P \text{ sqp } y$
 <proof>

end

end

theory *CLMM-Transformation* **imports** *CLMM-Description*

begin

6 CLMM transformations

6.1 CLMM pool refinement

Given a pool P and a (square root) price π , the refinement operation consists in defining a new grid (if necessary) in such a way that π is one of the bounds on the grid.

definition *refine* **where**

refine P $sqp = (\text{let } i = \text{lower-tick } P \text{ } sqp \text{ in}$
 $(\text{if } (\text{grd } P \text{ } i = sqp) \text{ then } P \text{ else}$
 $(\text{wedge } (\text{grd } P) \text{ } i \text{ } sqp, \text{ wedge } (lq \text{ } P) \text{ } i \text{ } (lq \text{ } P \text{ } i), \text{ wedge } (fee \text{ } P) \text{ } i \text{ } (fee \text{ } P \text{ } i))))$

lemma *refine-eq*:

assumes $i = \text{lower-tick } P \text{ } sqp$
 and $\text{grd } P \text{ } i = sqp$
shows $\text{refine } P \text{ } sqp = P$ $\langle \text{proof} \rangle$

lemma *refine-lq*:

assumes $i = \text{lower-tick } P \text{ } sqp$
 and $\text{grd } P \text{ } i \neq sqp$
 and $P' = \text{refine } P \text{ } sqp$
shows $lq \text{ } P' = \text{wedge } (lq \text{ } P) \text{ } i \text{ } (lq \text{ } P \text{ } i)$
 $\langle \text{proof} \rangle$

lemma *refine-fee*:

assumes $i = \text{lower-tick } P \text{ } sqp$
 and $\text{grd } P \text{ } i \neq sqp$
 and $P' = \text{refine } P \text{ } sqp$
shows $fee \text{ } P' = \text{wedge } (fee \text{ } P) \text{ } i \text{ } (fee \text{ } P \text{ } i)$
 $\langle \text{proof} \rangle$

lemma *refine-grd*:

assumes $i = \text{lower-tick } P \text{ } sqp$
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{grd } P \text{ } i \neq sqp$
shows $\text{grd } P' = \text{wedge } (\text{grd } P) \text{ } i \text{ } sqp$
 $\langle \text{proof} \rangle$

lemma *refine-grd-cong*:
 assumes $P1 = \text{refine } P \text{ sqp}$
 and $P2 = \text{refine } P' \text{ sqp}$
 and $\text{grd } P = \text{grd } P'$
shows $\text{grd } P1 = \text{grd } P2$
 $\langle \text{proof} \rangle$

lemma *refine-grd-arg-le*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $j \leq i$
shows $\text{grd } P' j = \text{grd } P j$
 $\langle \text{proof} \rangle$

lemma *refine-grd-arg-gt*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $\text{grd } P (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$
 and $i < j$
shows $\text{grd } P' (j+1) = \text{grd } P j$
 $\langle \text{proof} \rangle$

lemma *refine-grd-arg-Suc*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $\text{grd } P (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$
shows $\text{grd } P' (i+1) = \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *refine-fee-arg-le*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $j \leq i$
shows $\text{fee } P' j = \text{fee } P j$
 $\langle \text{proof} \rangle$

lemma *refine-fee-arg-gt*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $\text{grd } P (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$
 and $i < j$
shows $\text{fee } P' (j+1) = \text{fee } P j$
 $\langle \text{proof} \rangle$

lemma *refine-fee-arg-Suc*:
 assumes $i = \text{lower-tick } P \text{ sqp}$
 and $P' = \text{refine } P \text{ sqp}$
 and $\text{grd } P (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$

shows $fee\ P'\ (i+1) = fee\ P\ i$
 $\langle proof \rangle$

lemma *refine-lq-arg-le*:
assumes $i = lower_tick\ P\ sqp$
and $P' = refine\ P\ sqp$
and $grd\ P\ i \neq sqp$
and $j \leq i$
shows $lq\ P'\ j = lq\ P\ j$
 $\langle proof \rangle$

lemma *refine-lq-arg-gt*:
assumes $i = lower_tick\ P\ sqp$
and $P' = refine\ P\ sqp$
and $grd\ P\ i \neq sqp$
and $i < j$
shows $lq\ P'\ (j+1) = lq\ P\ j$
 $\langle proof \rangle$

lemma *refine-lq-arg-Suc*:
assumes $i = lower_tick\ P\ sqp$
and $P' = refine\ P\ sqp$
and $grd\ P\ i \neq sqp$
shows $lq\ P'\ (i+1) = lq\ P\ i$
 $\langle proof \rangle$

lemma *refine-on-grd*:
assumes $clmm_dsc\ P$
and $grd\ P\ i = sqp$
shows $refine\ P\ sqp = P$
 $\langle proof \rangle$

lemma *refine-encomp-at-grd*:
assumes $clmm_dsc\ P$
and $P' = refine\ P\ sqp$
and $grd\ P\ (lower_tick\ P\ sqp) = sqp$
shows $encomp_at\ (grd\ P')\ (grd\ P)\ j\ j$
 $\langle proof \rangle$

lemma *refine-encomp-at-arg-le*:
assumes $clmm_dsc\ P$
and $P' = refine\ P\ sqp$
and $i = lower_tick\ P\ sqp$
and $grd\ P\ i \neq sqp$
and $j \leq i$
shows $encomp_at\ (grd\ P')\ (grd\ P)\ j\ j$
 $\langle proof \rangle$

lemma *refine-encomp-at-arg-ge-Suc*:

assumes *clmm-dsc* P
and $P' = \text{refine } P \text{ sqp}$
and $i = \text{lower-tick } P \text{ sqp}$
and $\text{grd } P \ i \neq \text{sqp}$
and $i+1 \leq j$
and $0 < \text{sqp}$
shows *encomp-at* $(\text{grd } P') (\text{grd } P) \ j \ (j-1)$
 $\langle \text{proof} \rangle$

lemma *refine-finer-range*:
assumes *clmm-dsc* P
and $0 < \text{sqp}$
and $P' = \text{refine } P \text{ sqp}$
shows *finer-range* $(\text{grd } P') (\text{grd } P)$
 $\langle \text{proof} \rangle$

lemma *refine-finite-liq*:
assumes *finite-liq* P
and $P' = \text{refine } P \text{ sqp}$
shows *finite-liq* P'
 $\langle \text{proof} \rangle$

lemma *refine-clmm*:
assumes *clmm-dsc* P
and $0 < \text{sqp}$
and $P' = \text{refine } P \text{ sqp}$
shows *clmm-dsc* P'
 $\langle \text{proof} \rangle$

lemma *refine-lower-tick-idx*:
assumes *clmm-dsc* P
and $0 < \text{sqp}$
and $i = \text{lower-tick } P \text{ sqp}$
and $P' = \text{refine } P \text{ sqp}$
and $\text{grd } P \ (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$
shows $\text{lower-tick } P' \text{ sqp} = i+1$
 $\langle \text{proof} \rangle$

lemma *refine-ge-lower-tick-eq*:
assumes *clmm-dsc* P
and $0 < \text{sqp}$
and $i = \text{lower-tick } P \text{ sqp}'$
and $P' = \text{refine } P \text{ sqp}$
and $\text{grd } P \ (\text{lower-tick } P \text{ sqp}) \neq \text{sqp}$
and $\text{sqp} \leq \text{sqp}'$
and $\text{lower-tick } P \text{ sqp} = \text{lower-tick } P \text{ sqp}'$
shows $\text{lower-tick } P' \text{ sqp}' = i+1$
 $\langle \text{proof} \rangle$

lemma *refine-ge-lower-tick-gt*:
 assumes *clmm-dsc P*
 and $0 < sqp$
 and $sqp \leq sqp'$
 and $i = \text{lower-tick } P \ sqp'$
 and $P' = \text{refine } P \ sqp$
 and $\text{grd } P \ (\text{lower-tick } P \ sqp) \neq sqp$
 and $\text{lower-tick } P \ sqp < \text{lower-tick } P \ sqp'$
shows $\text{lower-tick } P' \ sqp' = i+1$
<proof>

lemma *refine-ge-lower-tick*:
 assumes *clmm-dsc P*
 and $0 < sqp$
 and $sqp \leq sqp'$
 and $i = \text{lower-tick } P \ sqp'$
 and $P' = \text{refine } P \ sqp$
 and $\text{grd } P \ (\text{lower-tick } P \ sqp) \neq sqp$
shows $\text{lower-tick } P' \ sqp' = i+1$
<proof>

lemma *refine-lower-tick*:
 assumes *clmm-dsc P*
 and $P' = \text{refine } P \ sqp$
 and $0 < sqp$
shows $\text{grd } P' \ (\text{lower-tick } P' \ sqp) = sqp$
<proof>

lemma *refine-finer-ranges*:
 assumes *clmm-dsc P*
 and $0 < sqp$
 and $P' = \text{refine } P \ sqp$
shows $\text{finer-ranges } (\text{grd } P') \ (\text{grd } P)$
<proof>

lemma *refine-coarse-idx-grd*:
 assumes *clmm-dsc P*
 and $P' = \text{refine } P \ sqp$
 and $\text{grd } P \ (\text{lower-tick } P \ sqp) = sqp$
shows $\text{coarse-idx } (\text{grd } P') \ (\text{grd } P) \ j = j$
<proof>

lemma *refine-coarse-idx-arg-le*:
 assumes *clmm-dsc P*
 and $P' = \text{refine } P \ sqp$
 and $i = \text{lower-tick } P \ sqp$
 and $\text{grd } P \ i \neq sqp$
 and $j \leq i$
 and $0 < sqp$

shows *coarse-idx* (*grd P'*) (*grd P*) *j* = *j*
 ⟨*proof*⟩

lemma *refine-coarse-idx-arg-gt*:
assumes *clmm-dsc P*
and $0 < sqp$
and $P' = \text{refine } P \text{ } sqp$
and $i = \text{lower-tick } P \text{ } sqp$
and $\text{grd } P \text{ } i \neq sqp$
and $i+1 \leq j$
shows *coarse-idx* (*grd P'*) (*grd P*) *j* = *j*−1
 ⟨*proof*⟩

lemma *refine-lq-idx-neg*:
assumes *clmm-dsc P*
and $0 < sqp$
and $P' = \text{refine } P \text{ } sqp$
and $\text{grd } P \text{ } (\text{lower-tick } P \text{ } sqp) \neq sqp$
shows $\text{lq } P' \text{ } j = \text{lq } P \text{ } (\text{pool-coarse-idx } P' \text{ } P \text{ } j)$
 ⟨*proof*⟩

lemma *refine-fee-idx-neg*:
assumes *clmm-dsc P*
and $0 < sqp$
and $P' = \text{refine } P \text{ } sqp$
and $\text{grd } P \text{ } (\text{lower-tick } P \text{ } sqp) \neq sqp$
shows $\text{fee } P' \text{ } j = \text{fee } P \text{ } (\text{pool-coarse-idx } P' \text{ } P \text{ } j)$
 ⟨*proof*⟩

lemma *refine-cst-fees*:
assumes $\bigwedge i. \text{fee } P \text{ } i = \text{phi}$
and $P' = \text{refine } P \text{ } sqp$
shows $\bigwedge i. \text{fee } P' \text{ } i = \text{phi}$
 ⟨*proof*⟩

lemma *refine-finer-neg*:
assumes *clmm-dsc P*
and $0 < sqp$
and $P' = \text{refine } P \text{ } sqp$
and $\text{grd } P \text{ } (\text{lower-tick } P \text{ } sqp) \neq sqp$
shows *finer-pool* $P' \text{ } P$
 ⟨*proof*⟩

lemma *refine-finer*:
assumes *clmm-dsc P*
and $0 < sqp$
and $P' = \text{refine } P \text{ } sqp$
shows *finer-pool* $P' \text{ } P$
 ⟨*proof*⟩

lemma *refine-nz-lq-sub*:
 assumes *clmm-dsc* P
 and $0 < sqp$
 and $P' = \text{refine } P \text{ } sqp$
shows $(\lambda j. \text{pool-coarse-idx } P' \text{ } P \text{ } j) \text{ ' } \text{nz-support } (lq \text{ } P') \subseteq$
 $\text{nz-support } (lq \text{ } P)$
 $\langle \text{proof} \rangle$

lemma *refine-nz-lq-ne*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{nz-support } (lq \text{ } P) \neq \{\}$
shows $\text{nz-support } (lq \text{ } P') \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *refine-nz-lq-emp*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{nz-support } (lq \text{ } P) = \{\}$
shows $\text{nz-support } (lq \text{ } P') = \{\}$
 $\langle \text{proof} \rangle$

lemma *refine-idx-min-eq*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{idx-min } (lq \text{ } P) \leq \text{lower-tick } P \text{ } sqp$
shows $\text{idx-min } (lq \text{ } P') = \text{idx-min } (lq \text{ } P)$
 $\langle \text{proof} \rangle$

lemma *refine-idx-min-Suc-eq*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{nz-support } (lq \text{ } P) \neq \{\}$
 and $\text{grd } P \text{ } (\text{lower-tick } P \text{ } sqp) \neq sqp$
 and $\text{lower-tick } P \text{ } sqp < \text{idx-min } (lq \text{ } P)$
shows $\text{idx-min } (lq \text{ } P') = \text{idx-min } (lq \text{ } P) + 1$
 $\langle \text{proof} \rangle$

lemma *refine-grd-min*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$
 and $\text{nz-support } (lq \text{ } P) \neq \{\}$
shows $\text{grd-min } P = \text{grd-min } P'$
 $\langle \text{proof} \rangle$

lemma *refine-idx-max-eq*:
 assumes *clmm-dsc* P
 and $P' = \text{refine } P \text{ } sqp$

and $\text{idx-max } (lq \ P) < \text{lower-tick } P \ sqp$
shows $\text{idx-max } (lq \ P') = \text{idx-max } (lq \ P)$
 $\langle \text{proof} \rangle$

lemma *refine-idx-max-Suc-eq*:
assumes *clmm-dsc* P
and $P' = \text{refine } P \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{grd } P \ (\text{lower-tick } P \ sqp) \neq sqp$
and $\text{lower-tick } P \ sqp \leq \text{idx-max } (lq \ P)$
shows $\text{idx-max } (lq \ P') = \text{idx-max } (lq \ P) + 1$
 $\langle \text{proof} \rangle$

lemma *refine-lower-tick-idx-max*:
assumes *clmm-dsc* P
and $0 < sqp$
and $P' = \text{refine } P \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $\text{lower-tick } P \ sqp \leq \text{idx-max } (lq \ P)$
shows $\text{lower-tick } P' \ sqp \leq \text{idx-max } (lq \ P')$
 $\langle \text{proof} \rangle$

lemma *refine-grd-max*:
assumes *clmm-dsc* P
and $P' = \text{refine } P \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{grd-max } P = \text{grd-max } P'$
 $\langle \text{proof} \rangle$

lemma *refine-quote-gross*:
assumes *clmm-dsc* P
and $P' = \text{refine } P \ sqp$
and $0 < sqp$
shows $\text{quote-gross } P' = \text{quote-gross } P$
 $\langle \text{proof} \rangle$

lemma *refine-nonzero-liq*:
assumes *clmm-dsc* P
and $\text{lower-tick } P \ sqp \leq i$
and $\text{grd } P \ (\text{lower-tick } P \ sqp) \neq sqp$
and $P' = \text{refine } P \ sqp$
and $L = lq \ P$
and $L' = lq \ P'$
shows $\{l. L' \ l \neq 0 \wedge i+1 < l\} = (\lambda i. i + 1) \ ` \ \{k. L \ k \neq 0 \wedge i < k\}$
 $\langle \text{proof} \rangle$

lemma *refine-pool-base-net-grd-eq*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$

and $P' = \text{refine } P \text{ } sqp$
and $0 < sqp$
and $sqp < \text{grd-max } P$
and $\text{grd } P (\text{lower-tick } P \text{ } sqp) \neq sqp$
and $sqp \leq sqp'$
shows $\text{base-net } P' \text{ } sqp' = \text{base-net } P \text{ } sqp'$
 $\langle \text{proof} \rangle$

lemma *refine-base-net-eq*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \text{ } P) \neq \{\}$
and $P' = \text{refine } P \text{ } sqp$
and $0 < sqp$
and $sqp < \text{grd-max } P$
and $sqp \leq sqp'$
shows $\text{base-net } P' \text{ } sqp' = \text{base-net } P \text{ } sqp'$
 $\langle \text{proof} \rangle$

6.2 CLMM pool restriction and slice

The restriction operation intuitively consists in deleting all the liquidity potentially available below the index provided as an argument.

definition *restrict-pool where*

$\text{restrict-pool } i \text{ } P =$
 $(\text{grd } P,$
 $(\lambda j. \text{ if } j < i \text{ then } 0 \text{ else } lq \text{ } P \text{ } j),$
 $(\lambda j. \text{ fee } P \text{ } j))$

lemma *restrict-pool-grd[simp]*:
shows $\text{grd } (\text{restrict-pool } i \text{ } P) = \text{grd } P$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-lower-tick*:
assumes $P' = \text{restrict-pool } i \text{ } P$
shows $\text{lower-tick } P \text{ } sqp = \text{lower-tick } P' \text{ } sqp$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-lt*:
assumes $j < i$
shows $lq (\text{restrict-pool } i \text{ } P) \text{ } j = 0$ $\text{fee } (\text{restrict-pool } i \text{ } P) \text{ } j = \text{fee } P \text{ } j$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-ge*:
assumes $i \leq j$
shows $lq (\text{restrict-pool } i \text{ } P) \text{ } j = lq \text{ } P \text{ } j$
 $\text{fee } (\text{restrict-pool } i \text{ } P) \text{ } j = \text{fee } P \text{ } j$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-lq-sub*:

assumes $P' = \text{restrict-pool } i \ P$
shows $\text{nz-support } (lq \ P') \subseteq \text{nz-support } (lq \ P)$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-finite-liq*:
assumes *finite-liq* P
and $P' = \text{restrict-pool } i \ P$
shows *finite-liq* P' $\langle \text{proof} \rangle$

lemma *restrict-pool-nz-liq*:
assumes *finite-liq* P
and $P' = \text{restrict-pool } i \ P$
and $i \leq \text{idx-max } (lq \ P)$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{nz-support } (lq \ P') \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-idx-max*:
assumes *finite-liq* P
and $P' = \text{restrict-pool } i \ P$
and $i \leq \text{idx-max } (lq \ P)$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{idx-max } (lq \ P) = \text{idx-max } (lq \ P')$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-clmm*:
assumes *clmm-dsc* P
and $P' = \text{restrict-pool } i \ P$
shows *clmm-dsc* P'
 $\langle \text{proof} \rangle$

lemma *restrict-pool-quote-gross-leq*:
assumes *mono* $(\text{grd } P)$
and $\text{sqp} \leq \text{grd } P \ i$
and $P' = \text{restrict-pool } i \ P$
shows $\text{quote-gross } P' \ \text{sqp} = 0 \ \langle \text{proof} \rangle$

lemma *restrict-pool-quote-gross*:
assumes *clmm-dsc* P
and $P' = \text{restrict-pool } j \ P$
and $0 < \text{sqp}$
and $j \leq \text{lower-tick } P \ \text{sqp}$
shows $\text{quote-gross } P \ \text{sqp} - \text{quote-gross } P \ (\text{grd } P \ j) = \text{quote-gross } P' \ \text{sqp}$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-base-net-eq*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{restrict-pool } i \ P$

and $i \leq \text{idx-max } (lq \ P)$
and $\text{grd } P \ i \leq \text{sqp}'$
shows $\text{base-net } P' \ \text{sqp}' = \text{base-net } P \ \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *restrict-pool-grd-min-le*:
assumes $\text{clmm-dsc } P$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{restrict-pool } i \ P$
and $i \leq \text{idx-max } (lq \ P)$
shows $i \leq \text{idx-min } (lq \ P')$
 $\langle \text{proof} \rangle$

definition *slice-pool where*
 $\text{slice-pool } P \ \text{sqp} = (\text{let } P' = \text{refine } P \ \text{sqp} \text{ in } \text{restrict-pool } (\text{lower-tick } P' \ \text{sqp}) \ P')$

lemma *slice-poolD*:
assumes $P'' = \text{refine } P \ \text{sqp}$
shows $\text{slice-pool } P \ \text{sqp} = \text{restrict-pool } (\text{lower-tick } P'' \ \text{sqp}) \ P''$
 $\langle \text{proof} \rangle$

lemma *slice-pool-clmm-dsc*:
assumes $\text{clmm-dsc } P$
and $0 < \text{sqp}$
and $P' = \text{slice-pool } P \ \text{sqp}$
shows $\text{clmm-dsc } P'$
 $\langle \text{proof} \rangle$

lemma *slice-pool-nz-liq*:
assumes $\text{clmm-dsc } P$
and $0 < \text{sqp}$
and $P' = \text{slice-pool } P \ \text{sqp}$
and $\text{lower-tick } P \ \text{sqp} \leq \text{idx-max } (lq \ P)$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{nz-support } (lq \ P') \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *slice-pool-tick-idx-max*:
assumes $\text{clmm-dsc } P$
and $0 < \text{sqp}$
and $P' = \text{slice-pool } P \ \text{sqp}$
and $\text{lower-tick } P \ \text{sqp} \leq \text{idx-max } (lq \ P)$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{lower-tick } P' \ \text{sqp} \leq \text{idx-max } (lq \ P')$
 $\langle \text{proof} \rangle$

lemma *slice-pool-nz-liq'*:
assumes $\text{clmm-dsc } P$
and $P' = \text{slice-pool } P \ \text{sqp}$

and $\text{nz-support } (lq \ P) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P$
shows $\text{nz-support } (lq \ P') \neq \{\}$
 $\langle \text{proof} \rangle$

lemma *slice-pool-idx-min*:
assumes *clmm-dsc* P
and $0 < sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{slice-pool } P \ sqp$
and $i = \text{lower-tick } P \ sqp$
and $i \leq \text{idx-max } (lq \ P)$
shows $i \leq \text{idx-min } (lq \ P')$
 $\langle \text{proof} \rangle$

lemma *slice-pool-grd-min*:
assumes *clmm-dsc* P
and $0 < sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{slice-pool } P \ sqp$
and $sqp < \text{grd-max } P$
shows $sqp \leq \text{grd-min } P'$
 $\langle \text{proof} \rangle$

lemma *slice-pool-grd-max*:
assumes *clmm-dsc* P
and $0 < sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{slice-pool } P \ sqp$
and $\text{lower-tick } P \ sqp \leq \text{idx-max } (lq \ P)$
shows $\text{grd-max } P = \text{grd-max } P' \ \langle \text{proof} \rangle$

lemma *slice-pool-grd-max'*:
assumes *clmm-dsc* P
and $\text{nz-support } (lq \ P) \neq \{\}$
and $P' = \text{slice-pool } P \ sqp$
and $0 < sqp$
and $sqp < \text{grd-max } P$
shows $\text{grd-max } P = \text{grd-max } P'$
 $\langle \text{proof} \rangle$

lemma *slice-pool-cst-fees*:
assumes *clmm-dsc* P
and $P' = \text{slice-pool } P \ sqp$
and $\bigwedge i. \text{fee } P \ i = \text{phi}$
shows $\bigwedge i. \text{fee } P' \ i = \text{phi}$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-gross-leq*:

assumes *clmm-dsc* P
and $0 < sqp$
and $sqp' \leq sqp$
and $P' = \text{slice-pool } P \ sqp$
shows $\text{quote-gross } P' \ sqp' = 0$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-gross*:

assumes *clmm-dsc* P
and $0 < sqp$
and $sqp \leq sqp'$
and $P' = \text{slice-pool } P \ sqp$
shows $\text{quote-gross } P' \ sqp' = \text{quote-gross } P \ sqp' - \text{quote-gross } P \ sqp$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-gross-max-eq*:

assumes *clmm-dsc* P
and $P' = \text{slice-pool } P \ sqp$
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P$
and $i = \text{lower-tick } P \ sqp$
and $\text{grd } P \ i = sqp$
shows $\text{quote-gross } P' \ (\text{grd-max } P') = \text{quote-gross } P \ (\text{grd-max } P) - \text{quote-gross } P \ sqp$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-gross-inv*:

assumes *clmm-dsc* P
and $0 < sqp$
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $sqp < \text{grd-max } P$
and $0 < y$
and $P' = \text{slice-pool } P \ sqp$
shows $\text{quote-gross } P' - \{y\} = \text{quote-gross } P - \{y + \text{quote-gross } P \ sqp\}$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-reach*:

assumes *clmm-dsc* P
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P$
and $0 < y$
and $P' = \text{slice-pool } P \ sqp$
shows $\text{quote-reach } P' \ y = \text{quote-reach } P \ (y + \text{quote-gross } P \ sqp)$
 $\langle \text{proof} \rangle$

lemma *slice-pool-base-net-eq*:

assumes *clmm-dsc* P
and *nz-support* $(lq\ P) \neq \{\}$
and $P' = \text{slice-pool } P\ sqp$
and $0 < sqp$
and $sqp < \text{grd-max } P$
and $sqp \leq sqp'$
shows $\text{base-net } P'\ sqp' = \text{base-net } P\ sqp'$
 $\langle \text{proof} \rangle$

lemma *slice-pool-base-net-slice*:
assumes *clmm-dsc* P
and *nz-support* $(lq\ P) \neq \{\}$
and $i = \text{lower-tick } P\ sqp$
and $P' = \text{slice-pool } P\ sqp$
and $sqp < \text{grd-max } P$
and $\text{grd } P\ i = sqp$
and $sqp' \leq sqp$
and $0 < sqp'$
shows $\text{base-net } P'\ sqp' = \text{base-net } P'\ sqp$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-swap-gt-zero*:
assumes *clmm-dsc* P
and *nz-support* $(lq\ P) \neq \{\}$
and $\text{grd } P\ (\text{lower-tick } P\ sqp2) = sqp2$
and $P' = \text{slice-pool } P\ sqp2$
and $sqp1 \leq sqp2$
and $0 < y$
and $0 < sqp1$
and $y + \text{quote-gross } P\ sqp2 \leq \text{quote-gross } P\ (\text{grd-max } P)$
shows $\text{quote-swap } P'\ sqp1\ y = \text{quote-swap } P\ sqp2\ y$
 $\langle \text{proof} \rangle$

lemma *slice-pool-quote-swap*:
assumes *clmm-dsc* P
and *nz-support* $(lq\ P) \neq \{\}$
and $\text{grd } P\ (\text{lower-tick } P\ sqp2) = sqp2$
and $P' = \text{slice-pool } P\ sqp2$
and $sqp1 \leq sqp2$
and $sqp2 < \text{grd-max } P$
and $0 \leq y$
and $0 < sqp1$
and $y + \text{quote-gross } P\ sqp2 \leq \text{quote-gross } P\ (\text{grd-max } P)$
shows $\text{quote-swap } P'\ sqp1\ y = \text{quote-swap } P\ sqp2\ y$
 $\langle \text{proof} \rangle$

6.3 CLMM pool join

The join operation is meant to define a pool P on which swap operations can be viewed as a combination of swap operations on its two arguments. We use the convention that the pool fee is 0 on ranges where there is no liquidity.

definition *pool-fee-join* **where**

pool-fee-join $P1\ P2\ i = \text{fee-union } (lq\ P1\ i)\ (lq\ P2\ i)\ (\text{fee } P1\ i)\ (\text{fee } P2\ i)$

lemma *pool-fee-join-com*:

shows *pool-fee-join* $P1\ P2\ i = \text{pool-fee-join } P2\ P1\ i$
 $\langle \text{proof} \rangle$

definition *joint-pools* **where**

joint-pools $P\ P1\ P2 \longleftrightarrow (\text{grd } P) = (\text{grd } P1) \wedge (\text{grd } P) = (\text{grd } P2) \wedge$
 $(\forall i. lq\ P\ i = lq\ P1\ i + lq\ P2\ i) \wedge$
 $(\forall i. \text{fee } P\ i = \text{pool-fee-join } P1\ P2\ i)$

definition *pool-join* **where**

pool-join $P1\ P2 =$
 $(\text{grd } P1, (\lambda i. lq\ P1\ i + lq\ P2\ i), (\lambda i. \text{pool-fee-join } P1\ P2\ i))$

lemma *joint-poolsI[intro]*:

assumes $\text{grd } P = \text{grd } P1$
and $\text{grd } P = \text{grd } P2$
and $\bigwedge i. lq\ P\ i = lq\ P1\ i + lq\ P2\ i$
and $\bigwedge i. \text{fee } P\ i = \text{pool-fee-join } P1\ P2\ i$
shows *joint-pools* $P\ P1\ P2\ \langle \text{proof} \rangle$

lemma *pool-join-joint*:

assumes $\text{grd } P1 = \text{grd } P2$
and $P = \text{pool-join } P1\ P2$
shows *joint-pools* $P\ P1\ P2\ \langle \text{proof} \rangle$

lemma *joint-pools-grids*:

assumes *joint-pools* $P\ P1\ P2$
shows $(\text{grd } P) = (\text{grd } P1)\ (\text{grd } P) = (\text{grd } P2)$
 $\langle \text{proof} \rangle$

lemma *joint-pools-lq*:

assumes *joint-pools* $P\ P1\ P2$
shows $lq\ P\ i = lq\ P1\ i + lq\ P2\ i$
 $\langle \text{proof} \rangle$

lemma *joint-pools-fee*:

assumes *joint-pools* $P\ P1\ P2$
shows $\text{fee } P\ i = \text{pool-fee-join } P1\ P2\ i$
 $\langle \text{proof} \rangle$

lemma *joint-pools-com*:
assumes *joint-pools* P $P1$ $P2$
shows *joint-pools* P $P2$ $P1$
 $\langle proof \rangle$

lemma *joint-pools-nz-liq-sub*:
assumes *joint-pools* P $P1$ $P2$
shows $nz\text{-support } (lq\ P) \subseteq nz\text{-support } (lq\ P1) \cup (nz\text{-support } (lq\ P2))$
 $\langle proof \rangle$

lemma *joint-pools-nz-liq-sup*:
assumes *joint-pools* P $P1$ $P2$
and $\bigwedge i. 0 \leq lq\ P1\ i$
and $\bigwedge i. 0 \leq lq\ P2\ i$
shows $nz\text{-support } (lq\ P1) \cup (nz\text{-support } (lq\ P2)) \subseteq nz\text{-support } (lq\ P)$
 $\langle proof \rangle$

lemma *joint-pools-nz-liq*:
assumes *joint-pools* P $P1$ $P2$
and $\bigwedge i. 0 \leq lq\ P1\ i$
and $\bigwedge i. 0 \leq lq\ P2\ i$
shows $nz\text{-support } (lq\ P1) \cup (nz\text{-support } (lq\ P2)) = nz\text{-support } (lq\ P)$
 $\langle proof \rangle$

lemma *clmm-joint-pools-nz-liq*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $nz\text{-support } (lq\ P1) \cup (nz\text{-support } (lq\ P2)) = nz\text{-support } (lq\ P)$
 $\langle proof \rangle$

lemma *joint-pools-finite-liq*:
assumes *finite-liq* $P1$
and *finite-liq* $P2$
and *joint-pools* P $P1$ $P2$
shows *finite-liq* P $\langle proof \rangle$

lemma *joint-pools-idx-min-min*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
and $nz\text{-support } (lq\ P1) \neq \{\}$
and $idx\text{-min } (lq\ P1) \leq idx\text{-min } (lq\ P2)$
shows $idx\text{-min } (lq\ P) = idx\text{-min } (lq\ P1)$
 $\langle proof \rangle$

lemma *joint-pools-idx-min*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$

and *joint-pools* P $P1$ $P2$
and *nz-support* $(lq\ P1) \neq \{\}$
and *nz-support* $(lq\ P2) \neq \{\}$
shows $idx-min\ (lq\ P) = \min\ (idx-min\ (lq\ P1))\ (idx-min\ (lq\ P2))$
 $\langle proof \rangle$

lemma *joint-pools-idx-max-max*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
and *nz-support* $(lq\ P2) \neq \{\}$
and $idx-max\ (lq\ P1) \leq idx-max\ (lq\ P2)$
shows $idx-max\ (lq\ P) = idx-max\ (lq\ P2)$
 $\langle proof \rangle$

lemma *joint-pools-idx-max*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
and *nz-support* $(lq\ P1) \neq \{\}$
and *nz-support* $(lq\ P2) \neq \{\}$
shows $idx-max\ (lq\ P) = \max\ (idx-max\ (lq\ P1))\ (idx-max\ (lq\ P2))$
 $\langle proof \rangle$

lemma *joint-pools-clmm-dsc*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows *clmm-dsc* P
 $\langle proof \rangle$

lemma *join-gross-fct*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $gross-fct\ (lq\ P)\ (fee\ P)\ i = gross-fct\ (lq\ P1)\ (fee\ P1)\ i +$
 $gross-fct\ (lq\ P2)\ (fee\ P2)\ i$
 $\langle proof \rangle$

lemma *quote-gross-join*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $quote-gross\ P\ x = quote-gross\ P1\ x + quote-gross\ P2\ x$
 $\langle proof \rangle$

lemma *quote-net-join*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$

and *joint-pools* P $P1$ $P2$
shows $\text{quote-net } P \ x = \text{quote-net } P1 \ x + \text{quote-net } P2 \ x$
 $\langle \text{proof} \rangle$

lemma *base-gross-join*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $\text{base-gross } P \ x = \text{base-gross } P1 \ x + \text{base-gross } P2 \ x$
 $\langle \text{proof} \rangle$

lemma *base-net-join*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $\text{base-net } P \ x = \text{base-net } P1 \ x + \text{base-net } P2 \ x$
 $\langle \text{proof} \rangle$

lemma *mkt-depth-join*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
shows $\text{mkt-depth } P \ x \ x' = \text{mkt-depth } P1 \ x \ x' + \text{mkt-depth } P2 \ x \ x'$
 $\langle \text{proof} \rangle$

lemma *joint-quote-gross-decomp*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $\text{grd-min } P \leq x$
and $0 \leq y$
and $y + \text{quote-gross } P \ x \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $x' = \text{quote-reach } P \ (y + \text{quote-gross } P \ x)$
and $y1 = \text{quote-gross } P1 \ x' - \text{quote-gross } P1 \ x$
and $y2 = \text{quote-gross } P2 \ x' - \text{quote-gross } P2 \ x$
shows $y = y1 + y2$
 $\langle \text{proof} \rangle$

lemma *joint-quote-gross-decomp'*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* P $P1$ $P2$
and $\text{nz-support } (\text{lq } P) \neq \{\}$
and $0 \leq y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $x' = \text{quote-reach } P \ y$
and $y1 = \text{quote-gross } P1 \ x'$
and $y2 = \text{quote-gross } P2 \ x'$

shows $y = y1 + y2$
 $\langle proof \rangle$

lemma *joint-base-net-decomp'*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* $P P1 P2$
and *nz-support* $(lq P) \neq \{\}$
and $0 \leq y$
and $y \leq \text{quote-gross } P (\text{grd-max } P)$
and $x' = \text{quote-reach } P y$
and $y1 = \text{base-net } P1 x'$
and $y2 = \text{base-net } P2 x'$
shows $\text{base-net } P x' = y1 + y2$
 $\langle proof \rangle$

lemma *joint-base-gross-decomp*:
assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and *joint-pools* $P P1 P2$
and *nz-support* $(lq P) \neq \{\}$
and $x \leq \text{grd-max } P$
and $0 \leq y$
and $y + \text{base-gross } P x \leq \text{base-gross } P (\text{grd-min } P)$
and $x' = \text{base-reach } P (y + \text{base-gross } P x)$
and $y1 = \text{base-gross } P1 x' - \text{base-gross } P1 x$
and $y2 = \text{base-gross } P2 x' - \text{base-gross } P2 x$
shows $y = y1 + y2$
 $\langle proof \rangle$

definition *join-pools* **where**
 $\text{join-pools } P1 P2 =$
 $(\text{grd } P1,$
 $(\lambda i. lq P1 i + lq P2 i),$
 $(\lambda i. \text{pool-fee-join } P1 P2 i))$

lemma *join-pools-grd[simp]*:
assumes $P = \text{join-pools } P1 P2$
shows $\text{grd } P = \text{grd } P1$ $\langle proof \rangle$

lemma *join-pools-lq[simp]*:
assumes $P = \text{join-pools } P1 P2$
shows $lq P i = lq P1 i + lq P2 i$
 $\langle proof \rangle$

lemma *join-pools-fee[simp]*:
assumes $P = \text{join-pools } P1 P2$
shows $\text{fee } P i = \text{pool-fee-join } P1 P2 i$
 $\langle proof \rangle$

lemma *join-joint-pools*:
assumes $\text{grd } P1 = \text{grd } P2$
shows $\text{joint-pools } (\text{join-pools } P1 \ P2) \ P1 \ P2$
 $\langle \text{proof} \rangle$

6.4 CLMM pool combination

definition *pool-comb* **where**
 $\text{pool-comb } P1 \ P2 \ \text{sqp} = (\text{let } P' = \text{refine } P1 \ \text{sqp} \text{ in}$
 $\text{pool-join } P' \ (\text{slice-pool } P2 \ \text{sqp}))$

lemma *pool-comb-joint*:
assumes $\text{grd } P1 = \text{grd } P2$
shows $\text{joint-pools } (\text{pool-comb } P1 \ P2 \ \text{sqp}) \ (\text{refine } P1 \ \text{sqp})$
 $(\text{slice-pool } P2 \ \text{sqp}) \ \langle \text{proof} \rangle$

lemma *pool-comb-refined-joint-nz-liq*:
assumes $\text{grd } P1 = \text{grd } P2$
and $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $P = \text{pool-comb } P1 \ P2 \ \text{sqp}$
and $\text{grd } P1 \ (\text{lower-tick } P1 \ \text{sqp}) = \text{sqp}$
shows $\text{nz-support } (\text{lq } P) = \text{nz-support } (\text{lq } P1) \cup$
 $(\text{nz-support } (\text{lq } (\text{slice-pool } P2 \ \text{sqp})))$
 $\langle \text{proof} \rangle$

lemma *pool-comb-joint-refined*:
assumes $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \ (\text{lower-tick } P1 \ \text{sqp}) = \text{sqp}$
shows $\text{joint-pools } (\text{pool-comb } P1 \ P2 \ \text{sqp}) \ P1$
 $(\text{slice-pool } P2 \ \text{sqp})$
 $\langle \text{proof} \rangle$

lemma *pool-comb-clmm-dsc*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $0 < \text{sqp}$
and $P3 = \text{pool-comb } P1 \ P2 \ \text{sqp}$
shows $\text{clmm-dsc } P3 \ \langle \text{proof} \rangle$

lemma *pool-comb-grd-min*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{nz-support } (\text{lq } P1) \neq \{\}$
and $\text{nz-support } (\text{lq } P2) \neq \{\}$
and $0 < \text{sqp}$

and $sqp < \text{grd-max } P2$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
shows $\text{grd-min } P = \min (\text{grd-min } P1) (\text{grd-min } (\text{slice-pool } P2 \ sqp))$
 $\langle \text{proof} \rangle$

lemma *pool-comb-le-grd-min*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{nz-support } (\text{lq } P1) \neq \{\}$
and $\text{nz-support } (\text{lq } P2) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P2$
and $\text{grd-min } P1 \leq sqp$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
shows $\text{grd-min } P = \text{grd-min } P1$
 $\langle \text{proof} \rangle$

lemma *pool-comb-grd-max*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{nz-support } (\text{lq } P1) \neq \{\}$
and $\text{nz-support } (\text{lq } P2) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P2$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
shows $\text{grd-max } P = \max (\text{grd-max } P1) (\text{grd-max } P2)$
 $\langle \text{proof} \rangle$

lemma *pool-comb-grd-max-slice*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{nz-support } (\text{lq } P1) \neq \{\}$
and $\text{nz-support } (\text{lq } P2) \neq \{\}$
and $0 < sqp$
and $sqp < \text{grd-max } P2$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
shows $sqp < \text{grd-max } P$
 $\langle \text{proof} \rangle$

lemma *pool-comb-quote-decomp*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 (\text{lower-tick } P1 \ sqp) = sqp$
and $0 < sqp$
and $sqp' \leq \text{grd-max } P$

and $P = \text{pool-comb } P1 \ P2 \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-gross } P \ sqp' = \text{quote-gross } P1 \ sqp' + \text{quote-gross } (\text{slice-pool } P2 \ sqp) \ sqp'$
 $\langle \text{proof} \rangle$

lemma *pool-comb-quote-le-slice*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \ (\text{lower-tick } P1 \ sqp) = sqp$
and $0 < sqp$
and $sqp' \leq sqp$
and $sqp \leq \text{grd-max } P$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-gross } P \ sqp' = \text{quote-gross } P1 \ sqp'$
 $\langle \text{proof} \rangle$

lemma *pool-comb-quote-diff-decomp*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \ (\text{lower-tick } P1 \ sqp) = sqp$
and $0 < sqp$
and $0 < sqp'$
and $0 < sqp1$
and $sqp' \leq \text{grd-max } P$
and $sqp1 \leq \text{grd-max } P$
and $P = \text{pool-comb } P1 \ P2 \ sqp$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-gross } P \ sqp' - \text{quote-gross } P \ sqp1 =$
 $\text{quote-gross } P1 \ sqp' - \text{quote-gross } P1 \ sqp1 +$
 $\text{quote-gross } (\text{slice-pool } P2 \ sqp) \ sqp' - \text{quote-gross } (\text{slice-pool } P2 \ sqp) \ sqp1$
 $\langle \text{proof} \rangle$

lemma *pool-comb-base-net-plus*:
assumes $\text{clmm-dsc } P1$
and $\text{clmm-dsc } P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \ (\text{lower-tick } P1 \ sqp2) = sqp2$
and $0 < sqp2$
and $0 < y$
and $y \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $P = \text{pool-comb } P1 \ P2 \ sqp2$
and $sqp' = \text{quote-reach } P \ y$
and $sqp' \leq sqp2$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{base-net } P \ sqp' = \text{base-net } P1 \ sqp' + \text{base-net } (\text{slice-pool } P2 \ sqp2) \ sqp'$

$\langle \text{proof} \rangle$

lemma *combo-quote-init1*:

assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \text{ (lower-tick } P1 \text{ sqp2)} = \text{sqp2}$
and $0 < \text{sqp2}$
and $P = \text{pool-comb } P1 \text{ } P2 \text{ sqp2}$
and $0 < y$
and $\text{nz-support (lq } P1) \neq \{\}$
and $\text{nz-support (lq } P2) \neq \{\}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $\text{sqp2} < \text{grd-max } P2$
and $\text{sqp1} \leq \text{sqp2}$

shows $\text{quote-gross } P \text{ sqp1} = \text{quote-gross } P1 \text{ sqp1}$

$\langle \text{proof} \rangle$

lemma *combo-remain-quote-eq*:

assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and $\text{grd } P1 = \text{grd } P2$
and $\text{grd } P1 \text{ (lower-tick } P1 \text{ sqp2)} = \text{sqp2}$
and $0 < \text{sqp2}$
and $P = \text{pool-comb } P1 \text{ } P2 \text{ sqp2}$
and $\text{nz-support (lq } P) \neq \{\}$
and $\text{nz-support (lq } P2) \neq \{\}$
and $0 < y$
and $0 < \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{sqp2} \leq \text{sqp}'$
and $\text{sqp1} \leq \text{sqp2}$
and $\text{rs1}' = \text{quote-reach } P2 (y - y1 + \text{quote-gross } P2 \text{ sqp2})$

shows $\text{quote-gross } P2 \text{ sqp}' = \text{quote-gross } P2 \text{ rs1}'$

$\langle \text{proof} \rangle$

lemma *comb-quote-gross-le*:

assumes *clmm-dsc* $P1$
and *clmm-dsc* $P2$
and $\text{grd } P1 = \text{grd } P2$
and $0 < \text{sqp}$
and $\text{sqp} < \text{grd-max } P$
and $0 < y$
and $y \leq \text{quote-gross } P \text{ sqp}$
and $y \leq \text{quote-gross } P (\text{grd-max } P)$
and $P = \text{pool-comb } P1 \text{ } P2 \text{ sqp}$

and $sqp' = \text{quote-reach } P \ y$
and $\text{nz-support } (lq \ P) \neq \{\}$
shows $\text{quote-gross } P1 \ sqp' = y$
 $\langle \text{proof} \rangle$

locale *combined-pools* =
fixes $P1 \ P2 \ P \ sqp2$
assumes $\text{cmb-P1: clmm-dsc } P1$
and $\text{cmb-P2: clmm-dsc } P2$
and $\text{cmb-grd-eq: } \text{grd } P1 = \text{grd } P2$
and $\text{cmb-on-grid: } \text{grd } P1 \ (\text{lower-tick } P1 \ sqp2) = sqp2$
and $\text{cmb-nz1: nz-support } (lq \ P1) \neq \{\}$
and $\text{cmb-nz2: nz-support } (lq \ P2) \neq \{\}$
and $\text{cmb-comb: } P = \text{pool-comb } P1 \ P2 \ sqp2$
and $\text{cmb-pos: } 0 < sqp2$
and $\text{cmb-max: } sqp2 < \text{grd-max } P2$

begin

lemma *combined-P-prop*:
shows $\text{clmm-dsc } P \ \text{nz-support } (lq \ P) \neq \{\}$
 $\langle \text{proof} \rangle$

lemmas $\text{cmb-props} = \text{cmb-P1} \ \text{cmb-P2} \ \text{cmb-grd-eq} \ \text{cmb-on-grid} \ \text{cmb-nz1} \ \text{cmb-nz2}$
 $\text{cmb-comb} \ \text{cmb-pos} \ \text{cmb-max} \ \text{combined-P-prop}$

lemma *combo-joint-quote-gross-decomp*:
assumes $0 < y$
and $0 < sqp1$
and $y + \text{quote-gross } P \ sqp1 \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $sqp' = \text{quote-reach } P \ (y + \text{quote-gross } P \ sqp1)$
and $y1 = \text{quote-gross } P1 \ sqp' - \text{quote-gross } P1 \ sqp1$
and $P'' = \text{slice-pool } P2 \ sqp2$
and $y2' = \text{quote-gross } P'' \ sqp' - \text{quote-gross } P'' \ sqp1$
shows $y = y1 + y2' \ y1 \leq y \ 0 \leq y1$
 $y1 + \text{quote-gross } P1 \ sqp1 \leq \text{quote-gross } P1 \ (\text{grd-max } P1)$
 $y2' \leq \text{quote-gross } P'' \ (\text{grd-max } P2)$
 $\langle \text{proof} \rangle$

lemma *combo-joint-quote-gross-leq-max*:
assumes $0 < y$
and $0 < sqp1$
and $y + \text{quote-gross } P \ sqp1 \leq \text{quote-gross } P \ (\text{grd-max } P)$
and $sqp' = \text{quote-reach } P \ (y + \text{quote-gross } P \ sqp1)$
and $y1 = \text{quote-gross } P1 \ sqp' - \text{quote-gross } P1 \ sqp1$
shows $y - y1 + \text{quote-gross } P2 \ sqp2 \leq \text{quote-gross } P2 \ (\text{grd-max } P2)$
 $\langle \text{proof} \rangle$

lemma *combo-joint-quote-gross-price-le*:

assumes $0 < y$
and $\text{grd-min } P1 \leq \text{sqp1}$
and $\text{sqp1} \leq \text{sqp2}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{rs1} = \text{quote-reach } P1 (y1 + \text{quote-gross } P1 \text{ sqp1})$
shows $\text{rs1} \leq \text{sqp}'$
 $\langle \text{proof} \rangle$

lemma *combo-joint-quote-gross-decomp-leq*:
assumes $0 < y$
and $0 < \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $P'' = \text{slice-pool } P2 \text{ sqp2}$
and $\text{sqp1} \leq \text{sqp2}$
and $y2' = \text{quote-gross } P'' \text{ sqp}'$
shows $y = y1 + y2' \text{ } y1 \leq y \text{ } 0 \leq y1$
 $y1 + \text{quote-gross } P1 \text{ sqp1} \leq \text{quote-gross } P1 (\text{grd-max } P1)$
 $y2' \leq \text{quote-gross } P'' (\text{grd-max } P2)$
 $\langle \text{proof} \rangle$

lemma *combo-quote-swap-slice-eq*:
assumes $0 < \text{sqp1}$
and $0 < y$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
shows $\text{quote-swap } P \text{ sqp1 } y = \text{quote-swap } P1 \text{ sqp1 } y1 +$
 $\text{quote-swap } (\text{slice-pool } P2 \text{ sqp2}) \text{ sqp1 } (y - y1)$
 $\langle \text{proof} \rangle$

lemma *combo-quote-swap-eq*:
assumes $0 < \text{sqp1}$
and $0 < y$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{sqp1} \leq \text{sqp2}$
shows $\text{quote-swap } P \text{ sqp1 } y = \text{quote-swap } P1 \text{ sqp1 } y1 +$
 $\text{quote-swap } P2 \text{ sqp2 } (y - y1)$
 $\langle \text{proof} \rangle$

lemma *comb-add-above-gt*:
assumes $0 < y$
and $0 < \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$

and $sqp' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $y1 < y$
and $sqp1 \leq sqp2$
shows $sqp2 < sqp'$
 $\langle \text{proof} \rangle$

lemma *comb-add-above-add-eq*:
assumes $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $rs1 = \text{quote-reach } P1 (y1 + \text{quote-gross } P1 \text{ sqp1})$
shows $\text{quote-gross } P1 \text{ sqp}' = \text{quote-gross } P1 \text{ rs1}$
 $\langle \text{proof} \rangle$

lemma *comb-add-above-add-eq2*:
assumes $0 < y$
and $\text{grd-min } P1 \leq sqp1$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $sqp' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $sqp1 \leq sqp2$
and $y1 < y$
and $rs1' = \text{quote-reach } P2 (y - y1 + \text{quote-gross } P2 \text{ sqp2})$
shows $\text{quote-gross } P2 \text{ sqp}' = \text{quote-gross } P2 \text{ rs1}'$
 $\langle \text{proof} \rangle$

lemma *combo-joint-rest-qty-slice*:
assumes $0 < y$
and $0 < sqp1$
and $sqp1 \leq sqp2$
and $y + \text{quote-gross } P \text{ sqp1} < \text{quote-gross } P (\text{grd-max } P)$
and $sqp' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $P'' = \text{slice-pool } P2 \text{ sqp2}$
shows $y - y1 = \text{quote-gross } P'' \text{ sqp}'$
 $\langle \text{proof} \rangle$

lemma *combo-joint-rest-qty*:
assumes $0 < y$
and $0 < sqp1$
and $sqp1 \leq sqp2$
and $y + \text{quote-gross } P \text{ sqp1} < \text{quote-gross } P (\text{grd-max } P)$
and $sqp' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $sqp2 \leq sqp'$
shows $y - y1 = \text{quote-gross } P2 \text{ sqp}' - \text{quote-gross } P2 \text{ sqp2}$
 $\langle \text{proof} \rangle$

lemma *combo-joint-rest-qty-le*:
assumes $0 < y$

and $0 < sqp1$
and $sqp1 \leq sqp2$
and $y + \text{quote-gross } P \text{ } sqp1 < \text{quote-gross } P \text{ } (grd-max \text{ } P)$
and $sqp' = \text{quote-reach } P \text{ } (y + \text{quote-gross } P \text{ } sqp1)$
and $y1 = \text{quote-gross } P1 \text{ } sqp' - \text{quote-gross } P1 \text{ } sqp1$
shows $y - y1 + \text{quote-gross } P2 \text{ } sqp2 \leq \text{quote-gross } P2 \text{ } (grd-max \text{ } P2)$
 $\langle proof \rangle$

lemma *combo-joint-rest-price-pos*:
assumes $0 < y$
and $grd-min \text{ } P1 \leq sqp1$
and $y + \text{quote-gross } P \text{ } sqp1 \leq \text{quote-gross } P \text{ } (grd-max \text{ } P)$
and $sqp' = \text{quote-reach } P \text{ } (y + \text{quote-gross } P \text{ } sqp1)$
and $y1 = \text{quote-gross } P1 \text{ } sqp' - \text{quote-gross } P1 \text{ } sqp1$
and $rs1' = \text{quote-reach } P2 \text{ } (y - y1 + \text{quote-gross } P2 \text{ } sqp2)$
and $y1 < y$
shows $0 < rs1'$
 $\langle proof \rangle$

lemma *combo-joint-quote-gross-price-le'*:
assumes $0 < y$
and $grd-min \text{ } P1 \leq sqp1$
and $sqp1 \leq sqp2$
and $y + \text{quote-gross } P \text{ } sqp1 \leq \text{quote-gross } P \text{ } (grd-max \text{ } P)$
and $sqp' = \text{quote-reach } P \text{ } (y + \text{quote-gross } P \text{ } sqp1)$
and $y1 = \text{quote-gross } P1 \text{ } sqp' - \text{quote-gross } P1 \text{ } sqp1$
and $y1 < y$
and $rs1' = \text{quote-reach } P2 \text{ } (y - y1 + \text{quote-gross } P2 \text{ } sqp2)$
shows $rs1' \leq sqp'$
 $\langle proof \rangle$

lemma *comb-add-above-price1-leq*:
assumes $0 < y$
and $grd-min \text{ } P1 \leq sqp1$
and $y + \text{quote-gross } P \text{ } sqp1 \leq \text{quote-gross } P \text{ } (grd-max \text{ } P)$
and $sqp' = \text{quote-reach } P \text{ } (y + \text{quote-gross } P \text{ } sqp1)$
and $y1 = \text{quote-gross } P1 \text{ } sqp' - \text{quote-gross } P1 \text{ } sqp1$
and $sqp1 \leq sqp2$
and $0 \leq y2$
and $y - y2 + \text{quote-gross } P2 \text{ } sqp2 \leq \text{quote-gross } P2 \text{ } (grd-max \text{ } P2)$
and $y2 < y1$
and $y1 < y$
and $rs1 = \text{quote-reach } P1 \text{ } (y1 + \text{quote-gross } P1 \text{ } sqp1)$
and $rs2 = \text{quote-reach } P1 \text{ } (y2 + \text{quote-gross } P1 \text{ } sqp1)$
shows $rs2 \leq rs1$
 $\langle proof \rangle$

lemma *comb-add-above-price2-geq*:
assumes $0 < y$

and $\text{grd-min } P1 \leq \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{sqp1} \leq \text{sqp2}$
and $0 \leq y2$
and $y - y2 + \text{quote-gross } P2 \text{ sqp2} \leq \text{quote-gross } P2 (\text{grd-max } P2)$
and $y2 < y1$
and $y1 < y$
and $\text{rs1}' = \text{quote-reach } P2 (y - y1 + \text{quote-gross } P2 \text{ sqp2})$
and $\text{rs2}' = \text{quote-reach } P2 (y - y2 + \text{quote-gross } P2 \text{ sqp2})$
shows $\text{rs1}' \leq \text{rs2}'$
 $\langle \text{proof} \rangle$

lemma *comb-add-above-price2-geq'*:

assumes $0 < y$
and $\text{grd-min } P1 \leq \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{sqp1} \leq \text{sqp2}$
and $0 \leq y2$
and $y - y1 + \text{quote-gross } P2 \text{ sqp2} \leq \text{quote-gross } P2 (\text{grd-max } P2)$
and $y1 < y2$
and $y2 \leq y$
and $\text{rs1}' = \text{quote-reach } P2 (y - y1 + \text{quote-gross } P2 \text{ sqp2})$
and $\text{rs2}' = \text{quote-reach } P2 (y - y2 + \text{quote-gross } P2 \text{ sqp2})$
shows $\text{rs2}' \leq \text{rs1}'$
 $\langle \text{proof} \rangle$

lemma *comb-add-above-price2-lt*:

assumes $0 < y$
and $\text{grd-min } P1 \leq \text{sqp1}$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$
and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$
and $\text{sqp1} \leq \text{sqp2}$
and $0 \leq y2$
and $y - y2 + \text{quote-gross } P2 \text{ sqp2} \leq \text{quote-gross } P2 (\text{grd-max } P2)$
and $y2 < y1$
and $y1 < y$
and $\text{rs2}' = \text{quote-reach } P2 (y - y2 + \text{quote-gross } P2 \text{ sqp2})$
shows $\text{sqp}' < \text{rs2}'$
 $\langle \text{proof} \rangle$

lemma *combo-joint-reached-price-pos*:

assumes $0 < y$
and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$
and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$

shows $0 < sqp'$ $\langle proof \rangle$

lemma *combo-joint-base-reached-eq*:

assumes $0 < y$
and $grd-min\ P1 \leq sqp1$
and $y + quote-gross\ P\ sqp1 \leq quote-gross\ P\ (grd-max\ P)$
and $sqp' = quote-reach\ P\ (y + quote-gross\ P\ sqp1)$
and $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $sqp1 \leq sqp2$
and $rs1 = quote-reach\ P1\ (y1 + quote-gross\ P1\ sqp1)$
shows $base-net\ P1\ sqp' = base-net\ P1\ rs1$
 $\langle proof \rangle$

lemma *combo-joint-base-reached-eq2*:

assumes $0 < y$
and $grd-min\ P1 \leq sqp1$
and $y + quote-gross\ P\ sqp1 \leq quote-gross\ P\ (grd-max\ P)$
and $sqp' = quote-reach\ P\ (y + quote-gross\ P\ sqp1)$
and $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $sqp1 \leq sqp2$
and $y1 < y$
and $rs1' = quote-reach\ P2\ (y - y1 + quote-gross\ P2\ sqp2)$
shows $base-net\ P2\ sqp' = base-net\ P2\ rs1'$
 $\langle proof \rangle$

lemma *quote-gross-price-eq1*:

assumes $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $rs1 = quote-reach\ P1\ (y1 + quote-gross\ P1\ sqp1)$
shows $quote-gross\ P1\ rs1 = y1 + quote-gross\ P1\ sqp1$
 $\langle proof \rangle$

lemma *quote-gross-price-eq2*:

assumes $0 \leq y2$
and $y2 + quote-gross\ P1\ sqp1 \leq quote-gross\ P1\ (grd-max\ P1)$
and $rs2 = quote-reach\ P1\ (y2 + quote-gross\ P1\ sqp1)$
shows $quote-gross\ P1\ rs2 = y2 + quote-gross\ P1\ sqp1$
 $\langle proof \rangle$

end

6.5 Optimality result on quote tokens

When the fees in two pools are constant and equal, swapping a given amount of quote tokens in their combination permits to determine the optimal quantities of quote tokens to swap in each individual pool.

locale *combined-pools-cst-fee = combined-pools* +

fixes ϕ

assumes $fee1: \forall i. fee\ P1\ i = \phi$

and $fee2: \forall i. fee\ P2\ i = \phi$

begin

lemma *fee-props*:

shows $0 \leq \text{phi } \text{phi} < 1$ $\langle \text{proof} \rangle$

lemma *quote-swap-opt-above*:

assumes $0 < y$

and $\text{grd-min } P1 \leq \text{sqp1}$

and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$

and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$

and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$

and $\text{sqp1} \leq \text{sqp2}$

and $0 \leq y2$

and $y - y2 + \text{quote-gross } P2 \text{ sqp2} \leq \text{quote-gross } P2 (\text{grd-max } P2)$

and $y2 < y1$

and $y1 < y$

shows $\text{quote-swap } P1 \text{ sqp1 } y2 + \text{quote-swap } P2 \text{ sqp2 } (y - y2) \leq \text{quote-swap } P$
 $\text{sqp1 } y$

$\langle \text{proof} \rangle$

lemma *quote-swap-opt-above'*:

assumes $0 < y$

and $\text{grd-min } P1 \leq \text{sqp1}$

and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$

and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$

and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$

and $\text{sqp1} \leq \text{sqp2}$

and $y2 + \text{quote-gross } P1 \text{ sqp1} \leq \text{quote-gross } P1 (\text{grd-max } P1)$

and $0 \leq y - y2$

and $y1 < y2$

and $y2 \leq y$

shows $\text{quote-swap } P1 \text{ sqp1 } y2 + \text{quote-swap } P2 \text{ sqp2 } (y - y2) \leq \text{quote-swap } P$
 $\text{sqp1 } y$

$\langle \text{proof} \rangle$

lemma *combo-slice-no-addition2*:

assumes $0 < y$

and $y + \text{quote-gross } P \text{ sqp1} \leq \text{quote-gross } P (\text{grd-max } P)$

and $\text{sqp}' = \text{quote-reach } P (y + \text{quote-gross } P \text{ sqp1})$

and $y1 = \text{quote-gross } P1 \text{ sqp}' - \text{quote-gross } P1 \text{ sqp1}$

and $\text{sqp1} \leq \text{sqp2}$

and $y1 = y$

and $0 \leq y2$

and $y2 \leq y$

and $y - y2 + \text{quote-gross } P2 \text{ sqp2} \leq \text{quote-gross } P2 (\text{grd-max } P2)$

and $y1 \neq y2$

and $P'' = \text{slice-pool } P2 \text{ sqp2}$

shows $\text{quote-gross } P'' \text{ sqp}' = 0$

$\langle proof \rangle$

lemma *quote-swap-opt-below*:

assumes $0 < y$
and $grd-min\ P1 \leq sqp1$
and $y + quote-gross\ P\ sqp1 \leq quote-gross\ P\ (grd-max\ P)$
and $sqp' = quote-reach\ P\ (y + quote-gross\ P\ sqp1)$
and $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $sqp1 \leq sqp2$
and $y1 = y$
and $0 \leq y2$
and $y2 \leq y$
and $y - y2 + quote-gross\ P2\ sqp2 \leq quote-gross\ P2\ (grd-max\ P2)$
and $y1 \neq y2$
shows $quote-swap\ P1\ sqp1\ y2 + quote-swap\ P2\ sqp2\ (y - y2) \leq quote-swap\ P\ sqp1\ y$
 $\langle proof \rangle$

lemma *quote-swap-optimum'*:

assumes $0 < y$
and $grd-min\ P1 \leq sqp1$
and $y + quote-gross\ P\ sqp1 \leq quote-gross\ P\ (grd-max\ P)$
and $sqp' = quote-reach\ P\ (y + quote-gross\ P\ sqp1)$
and $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $sqp1 \leq sqp2$
and $0 \leq y2$
and $y2 \leq y$
and $y2 + quote-gross\ P1\ sqp1 \leq quote-gross\ P1\ (grd-max\ P1)$
and $y - y2 + quote-gross\ P2\ sqp2 \leq quote-gross\ P2\ (grd-max\ P2)$
and $y1 \neq y2$
shows $quote-swap\ P1\ sqp1\ y2 + quote-swap\ P2\ sqp2\ (y - y2) \leq quote-swap\ P\ sqp1\ y$
 $\langle proof \rangle$

lemma *quote-swap-optimum*:

assumes $0 < y$
and $0 < sqp1$
and $y + quote-gross\ P\ sqp1 \leq quote-gross\ P\ (grd-max\ P)$
and $sqp' = quote-reach\ P\ (y + quote-gross\ P\ sqp1)$
and $y1 = quote-gross\ P1\ sqp' - quote-gross\ P1\ sqp1$
and $sqp1 \leq sqp2$
and $grd-min\ P1 \leq sqp2$
and $0 \leq y2$
and $y2 \leq y$
and $y2 + quote-gross\ P1\ sqp1 \leq quote-gross\ P1\ (grd-max\ P1)$
and $y - y2 + quote-gross\ P2\ sqp2 \leq quote-gross\ P2\ (grd-max\ P2)$
and $y1 \neq y2$
shows $quote-swap\ P1\ sqp1\ y2 + quote-swap\ P2\ sqp2\ (y - y2) \leq quote-swap\ P\ sqp1\ y$

$\langle proof \rangle$

end

end

References

- [1] M. Echenim, E. Gobet, and A.-C. Maurice. Uniswap v3: im-permanent loss modeling and swap fees asymptotic analysis, Sept. 2025. Available at https://hal.science/hal-04214315v3/file/Article_IL_Uniswapv3_revision_HAL.pdf.